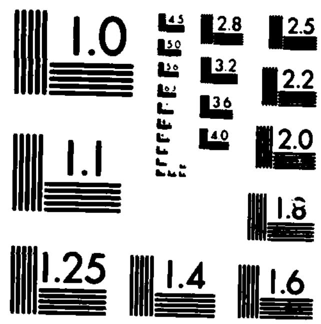
AD-A163 976 AN ANALYSIS OF THE ACCESIBILITY OF EARTH-APPROACHING ASTEROIDS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING P W SOMERS DEC 85 1/2

UNCLASSIFIED AFIT/GSO/AA/85D-1 F/G 22/3 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

AN ANALYSIS OF THE

ACCESSIBILITY OF EARTH-APPROACHING ASTEROIDS

THESIS

Philip W. Somers
Major, Canadian Armed Forces

AFIT/GSO/AA/85D-1

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GSO/AA/85D-1

AN ANALYSIS OF THE

ACCESSIBILITY OF EARTH-APPROACHING ASTEROIDS

THESIS

Philip W. Somers
Major, Canadian Armed Forces

AFIT/GSO/AA/85D-1

AFIT/GSO/AA/85D-1

AN ANALYSIS OF
THE ACCESSIBILITY OF EARTH-APPROACHING ASTEROIDS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Space Operations

Philip W. Somers, B.S.
Major, Canadian Armed Forces

December 1985

Preface

The purpose of this thesis was to demonstrate the practicality of conducting an analysis of the accessibility of Earth-approaching asteroids on a micro-computer. The work for this thesis combined aspects of astronomy and computers, in which my current interests are, respectively, asteroids and the Turbo PASCAL programming language. A reasonably ambitious computer program to integrate these interests into a practical tool for meaningful analyses seemed a worthwhile challenge.

The computer program calculates the change in velocity that must be imparted to a spacecraft to transfer it from an orbit around the Earth to an orbit around an asteroid. Although the program will work for any asteroid, only Earth-approaching asteroids were analyzed. Particular attention was given to the three Earth-approaching asteroids that were discovered from January to November 1985. Input to the program is a data file containing the classical orbital elements of the Earth and the asteroids. Output is via plots and tables that show the velocity changes required and the launch opportunities that will be available up to the year 2020 to achieve these intercept trajectories.

As a relative novice to orbital mechanics, I am indebted to my thesis advisor Dr. William Wiesel for his very knowledgeable guidance throughout this experience. I was fortunate to find an advisor with a previous academic interest in asteroids and I hope that I may have somewhat rekindled that interest.

I must also acknowledge an important contribution to my thesis from Mr. Neal Hulkower, whose work is often cited in this thesis. At a

critical stage in this work, a couple of long-distance telephone conversations with Mr. Hulkower firmly established the way to proceed.

To my wife and two young children go my sincere appreciation for the understanding and support throughout these last 18 months of academia. Now maybe the kids can play with the computer again.

Philip W. Somers

Table of Contents

Page

List of Figures

List of Tables

AFIT/GSO/AA/85D-1

Abstract

The purpose of this paper was to analyze the accessibility of Earth-approaching asteroids using a computer program that was practical to run on a micro-computer. This analysis employs techniques that can easily be adapted to find optimal trajectories for a variety of orbital intercept applications.

The mathematical analysis was adapted from recently-developed algorithms that were designed to run on main frame computers using extensive software libraries and data resources. The computer program developed for this paper was designed to operate on an IBM PC equipped with an 8087 math co-processor chip. Programming was done in Turbo PASCAL Version 3.0 which supports the 8087 mathematical capabilities. The program was designed to be self-contained except for data files of orbital elements. The program was also designed to operate efficiently and quickly while retaining much of the accuracy found on the main frame implementations. Only nonperturbed Keplerian motion was modelled. Every effort was made to ensure the program was as flexible as possible. Any object in the solar system in heliocentric orbit can be used as either the departure or arrival body. Orbital element data files are included for all the planets, several periodic comets, all the recently-discovered Earth-approaching asteroids, and many of the main belt asteroids. This flexibility permits not only rendezvous missions to be calculated, but can just as easily handle fly-by trajectories and return-to-Earth missions.

Extensive testing of the algorithms was conducted on all of the Earth-approaching asteroids discovered since 1982, and on several other Earth-approaching asteroids that have particularly-accessible orbits. Tabular and graphical data was formatted and displayed so as to be highly readable and practical to the user. Extensive use was made of the concept of Prime Rib curves to plot global optimum trajectory changes of velocity for all combinations of true anomalies of the departure and arrival planets. Validation of the program was done by comparison of the output to published results from calculations done on main frame computers, and by comparison of ephemerides with those published in the Astronomical Almanac. The results demonstrate the practicality of quickly and accurately analyzing the accessibility of Earth-approaching asteroids on a micro-computer.

AN ANALYSIS OF THE ACCESSIBILITY OF EARTH-APPROACHING ASTEROIDS

I. Introduction

A rendezvous mission to any of the 3500 known asteroids is possible, but the easiest and most practical missions would be to the Earth-approaching asteroids. However, no interplanetary probes have ever travelled to an asteroid. All of the information currently known about these minor planets has come from ground-based, or near-Earth-orbit sensors. Probes such as Pioneer and Voyager have gone to most of the planets. Spacecraft have landed on Mars and Venus, and flyby missions have photographed Mercury, Jupiter and Saturn. However, asteroids remain totally unexplored. Most asteroids are much closer to the Earth than Jupiter or Saturn, and many are easier to reach than any of the planets. In fact, some projected trips to land on asteroids and return to Earth would require a smaller expenditure of energy than a similar trip to the Moon. Of the 82 known Earth-approaching asteroids, 28 have been discovered in the 1980s, kindling a renewed interest in a rendezvous mission.

Algorithms have been developed to determine the accessibility (the energy requirements and the frequency of launch opportunities) of asteroids. To ensure accuracy and timeliness of the calculations, these algorithms have been implemented on large main frame computers with extensive software and data libraries. A practical implementation is needed that can be run on more modest computing facilities. With the many recent discoveries of Earth-approaching asteroids (10:321-328; 21:1-19), and the renewed interest in asteroid exploration, a

microcomputer implementation would greatly improve the utility of these algorithms. Accessibility information would then be available to many users, for the asteroids in which they had an interest. This information would also be widely available for newly-discovered asteroids as soon as the orbital elements were known.

II. Background

Many reasons have been offered to justify a rendezvous with an asteroid. Some scientists believe that the asteroids are made of very primitive material largely undisturbed since the formation of the solar system. Important scientific research could be accomplished with even a tiny sample of this material. Due to the large number of known asteroids, there is expected to be a correspondingly-large variety to the composition of their material. According to Stancadi and Soldner (18:1), the principle goal of missions to asteroids would be to determine chemical composition and structure to provide data for the study of the early history of the solar system. Due to the extremely low gravitational forces, several factors that cause chemical changes in the materials of the large planets are not a factor on asteroids. With no atmosphere, there is no wind or rain erosion or surface decomposition. Internal gravitational pressures are too low to produce hot liquid cores, except possibly to some degree in a few of the largest asteroids. The processes of heating, melting and differentiation of materials and elements that have radically changed terrestrial planets probably have not occurred with asteroids (23:434). Small size and low gravity make the asteroids small targets for impacts by crater-producing objects. However, they have no atmosphere to protect them from micrometeorites and must suffer constant bombardment. The effects of this bombardment would probably be confined to a thin layer on the surface. Beneath this thin layer, the material is expected to be more volatile than that of the inner planets because this material probably has formed at temperatures below 400 degrees Kelvin (23:437). There is some evidence that some asteroids

did not form in the asteroid belt, but at much greater distances from the sun as comets. If cometary ice is trapped in the interiors, there may be an outgassing of water vapor or other gasses (23:437).

The only data that is now available on the chemical composition of asteroids is from ground-based observations. By assuming that the surfaces of the asteroids are composed of common cosmic materials, in approximately the same proportions, spectral studies can predict the likely composition of individual asteroids (16:25-40). However, dark and/or opaque material remains hidden from spectroscopic study. The only feasible method of validating these remote observations is to compare the results to the composition of fragments of meteorites found on the Earth. Occasionally, through analyses of impact craters and photographic tracks of meteorites, it is possible to associate them with an asteroid or comet. Then a spectroscopic comparison can be made. Just as likely, a spectroscopic comparison can infer the origin of a meteorite without any other supporting evidence. Unlike the Moon, where samples returned from the surface provide positive spectroscopic proofing of remote observations, asteroids lack any confirmed comparisons. Therefore, analysis of surface materials remains an uncertain proposition.

The exact origin of meteorites that may have come from asteroids is of interest for other reasons. In 1890, the Farmington meteroite fell in Kansas. Although no photographic evidence was available, numerous newspaper and eyewitness reports enabled its orbit to be reconstructed. Analysis revealed that the meteorite was in a small Earth orbit prior to impact. Further back-plotting of the orbit associated the meteorite with either the asteroid Apollo, Hermes or Cerebus (15:234). Direct comparison of the L-chondrite material of the Farmington meteorite with material

from these or similar asteroids would be a break-through. Besides the obvious chemical-related answers, invaluable information could be obtained on the accuracy and reliability of some of the orbital paths, and the methods for establishing the origins of meteorites. Some researchers (22:54) believe that most of the meteorites falling on the Earth are fragments of high-eccentricity Apollo asteroids that have collided with other objects as they pass through the heavily-populated asteroid belt. However, no direct evidence is available to confirm this theory.

The most likely asteroids to have originated as comets are those whose orbits have a high eccentricity and/or a high inclination. Hahn and Rickman (9:420) tabulate 14 asteroids (ranging from 1036 Ganymed to recently-discovered 1984 BC) which have eccentricities greater than 0.47 and inclinations up to 34.57 degrees. Cockrane and Barker (4:296) contend that most of the numbered asteroids have eccentricities lower than typical comets, and have diameters that are too large. However, the Aten, Apollo and Amor class asteroids are in orbits much more closely matching comets. Many of these asteroids have diameters of five kilometers or less in diameter, which is typical of comets. Cockrane and Barker point out that the recently-discovered asteroid 1983 TB is in a "cometary" orbit that is almost identical to the orbit of 19 Geminid meteors. Spectral studies of 1983 TB could not detect any gas emission but could not conclusively deny cometary origin (4:297). Even if some asteroids were found to have a comet-like composition, there remains the orbital dynamics question of how sufficient numbers of comets were perturbed into asteroid orbits. Apollo-Amor asteroids typically have aphelion distances well inside the orbit of Jupiter. Periodic comets typically have aphelion distances well beyond the orbit of Jupiter.

Yamada (25:459) hypothesizes that the orbits of Apollo-Amor asteroids could be perturbed by encounters with the Earth and Venus to reduce the large aphelion distances. A positive link between comets and asteroids (such as a material sample) would promote a re-think of the possible mechanisms of these orbital perturbations. Farquhar and Dunham (6:1) describe realistic trajectories to return samples from Halley's comet. While it is too late for these missions, they do portray a scenario that could be used with other comets. The lowest cost missions (due to lowest energy requirements) are fly-by as opposed to rendezvous. A proposed mission would collect cometary dust during the high speed fly-by and return to a Shuttle-accessible Earth orbit using atmospheric braking. While this technique could not return samples from an asteroid, it does provide for part of the puzzle with material from a comet.

Besides the scientific interest in the chemical composition of asteroids, there are economic interests. If just one asteroid could be explored, the baseline data could be used to determine the expected mineral content of similar asteroids. Analysis of light from asteroids has already led scientists to speculate on mineral content (23:434). Asteroids with similar reflectivity and spectral light could contain similar minerals. If valuable minerals were found, it might be more economical (due to the low surface gravity) to mine an asteroid than to mine the Moon. If strategically-important minerals, such as titanium or uranium were found, asteroid exploration could have significant military implications.

It is possible that the orbit of a small asteroid could be deliberately altered. Since many of the recently-discovered asteroids are very small, often less than 500 meters in diameter (11:44), it is

conceivable that an asteroid could be manipulated as a weapon. It may even be necessary to defend against a random asteroid impact (22:54). Wetherill calculated that about once every 100 years, an asteroid about one kilometer in diameter can be expected to pass nearer to the Earth than the distance of the Moon. Such a body collides with the Earth about every 250,000 years. The energy released by such an impact would equal the yield of 10,000 10-megaton hydrogen bombs. However, the probability of an asteroid actually hitting the Earth is very small (8:171). Despite the remote possibility of a collision, the potential consequences may warrant at least a cursory look at the options. French and Hulkower (8:167) refer to three methods of avoiding a collision. The orbit of the asteroid could be altered by explosive devices, or by mass drivers accelerating asteroidal material to produce reaction thrust. For a small asteroid, total destruction might be possible. In the unlikely event that an asteroid were detected on a collision course with Earth, current knowledge about asteroids would make it very difficult to mount one of these defenses. The smaller Earth-approaching asteroids with diameters in the order of one kilometer are the least understood, but of course are the best candidates for collisions. No accurate size or mass data is available so the magnitude of the task to deflect such an object is based on inaccurate information. Composition of the asteroid would be critical if the mass driver or total destruction options were contemplated. For Earth-approaching asteroids, observational data to determine composition is scant or nonexistent (8:167). Unfortunately, if information and data needed to deflect an asteroid is ever required, sufficient time to obtain this information probably will not be available.

Further knowledge of the size and mass of asteroids is being gained by observing their interaction with large planets. Williams (24:1) was able to achieve significant mass determinations of the three largest asteroids by observing the perturbations they produced in the orbit of Mars. About three dozen other asteroids perturb Mars by more than five meters but ranging data is not sufficiently precise to produce accurate mass determinations. Ranging data available from Viking spacecraft can be used to determine the motion of the inner planets against an inertial frame of reference. A major objective of this ranging is to check on the constancy of the gravitational constant. However, for further Mars missions to contribute to this check, more accurate information on the sizes and densities of asteroids will be necessary (24:7). The accuracies required can only be determined accurately by direct measurement during fly-by or, preferably, rendezvous missions.

With the renewed interest in the Earth-approaching asteroids, there has been a corresponding increased interest in the orbital mechanics required to plan and support any possible rendezvous mission. Most of the capability of conducting meaningful analyses of intercept trajectories and long term projections has resided on large mainframe computers. With the advent of powerful microcomputers, and the development of new algorithms to analyze trajectories, accurate and timely analyses are now possible using modest computing resources.

III. Objectives

The overall objective of this paper is to demonstrate a computerized analysis of the accessibility of Earth-approaching asteroids with particular emphasis on the three most accessible asteroids, and those that have been discovered from January to November 1985.

To accomplish the overall objective, the sub-objectives are:

1. To determine which asteroids have been discovered during the period of interest. (Earth-approaching asteroids have recently been discovered at the rate of between three and eight per year. As well, previous discoveries are occasionally added or deleted from the list of these asteroids as the orbital parameters are refined by subsequent observations.)

2. To accurately calculate the orbits of the Earth and the selected asteroids, and the intercept trajectories.

3. To determine the accessibility of these asteroids.

4. To compare the accessibility of these asteroids with previous discoveries.

5. To compare the capability of the computer programs with results generated by main frame computer facilities.

6. To extend the current knowledge of the accessibility of Earth-approaching asteroids.

IV. Theory

Classification of Asteroids

To be designated as Earth-approaching, asteroids obviously must approach the Earth. The Apollo class asteroids are in orbits very similar to the orbit of the Earth, and occasionally, the orbital paths come close to intersecting. A new and significant asteroid discovery was made on a photograph taken on the night of 27/28 February 1982 at the Palomar Mountain Observatory. The asteroid, designated as 1982 DB, passed within 4.08 million kilometers of the Earth about one month before its discovery. An analysis of the orbit of 1982 DB revealed that it was an Apollo asteroid (an asteroid whose orbit crosses the Earth's orbit). This discovery was particularly significant because the asteroid had passed nearer the Earth than any other asteroid since 1976, and its orbit made it the easiest of all known asteroids to reach with a probe. A detailed analysis of intercept opportunities from 1982 to 2002 revealed that not only is 1982 DB easy to reach, but that two-way mission opportunities occur almost yearly (11:42).

Besides the Apollo asteroids which cross the Earth's orbit, there are two other classes of asteroids that occasionally approach the Earth. Aten asteroids orbit the sun just inside the Earth's orbit, and Amor asteroids orbit just outside. All three classes could include asteroids that would provide favorable rendezvous opportunities. Collectively, these asteroids are known as AAA (Apollo-Aten-Amor) or Earth-approaching asteroids.

> "An asteroid is said to be Earth-approaching if its distance at perihelion, q, is within 1.3 AU of the sun, thereby admitting

the possibility of passing within a few tenths of an AU of Earth. These bodies are divided into three families according to q and semimajor axis, a:

Family	Orbital Characteristics
Aten	q <= 1.02, a <= 1
Apollo	q <= 1.02, a > 1
Amor	1.02 <= q <= 1.3 " (18:1)

The "family" names are derived from a representative asteroid in each family whose orbit satisfies these conditions. Nominal orbital parameters of the AAA namesakes for epoch 12h 00 Dec 1985 (14:3) are:

Number	Name	q	a	e	i	AN	AP	M
2026	ATEN	.79	.97	.18	19	108	148	12
1862	APOLLO	.65	1.47	.56	6	35	285	326
1221	AMOR	1.09	1.92	.43	12	171	26	50

The Palomar Planet-Crossing Asteroid Survey has discovered several of these Earth-approaching asteroids since 1982. However, many of the orbital parameters are still only approximations, and only a few of these new asteroids have yet been given an official catalog number. Until they have been observed on several different orbits, and their orbital parameters accurately determined, they will be known only by the year of discovery followed by a two-letter serial code. Despite this uncertainty, the initial orbital parameters are generally sufficiently accurate to calculate how easily and how often a rendezvous could be made.

Optimal Trajectories

Mathematical techniques have been devised to calculate the energy required to rendezvous with a given asteroid at a given time. However, at a different time, the intercept geometry can be vastly different. It may not be at all apparent whether or not this new geometry represents a better or worse intercept opportunity. Each instant in time can be

analyzed directly and accurately, but no direct method has yet been devised to find the optimum intercept opportunities. It is currently believed that no set of equations can be found that will directly solve this problem, and no attempt will be made in this paper to do so.

Techniques have recently been developed to indirectly solve the minimum energy intercept problem (17:1-5; 12:458-461). These techniques, practical only when implemented on a computer, require that intercept energy be calculated for every conceivable position of an asteroid in its orbit for every conceivable position of the Earth in its orbit. The total number of calculations required depends on the desired resolution in the answer. If the positions of each body were taken each degree around its 360 degree orbit, a total of 129,600 (360 x 360) calculations would be required for each asteroid. Each of these calculations is far from trivial. For every intercept opportunity, the orbital position of the Earth and the asteroid must be calculated, as well as the satellite trajectory near the Earth, and the interplanetary trajectory under the influence of the sun. Parts of these calculations cannot be solved in closed form, but require computer iterative techniques to get an approximate answer. Because of the large number of calculations involved, even a computer can spend long hours trying to find the answers.

Fortunately, for the purpose of the analyses conducted in this paper, only a few of the 3500 known asteroids are classified as Earth-approaching. In 1983, McFadden listed 36 Amors, 34 Apollos and 3 Atens (16:25). Many of these 73 asteroids are obviously less accessible than two that have received recent attention. Detailed calculations have been done for 1943 Anteros, and 1982 DB. The latter was found to be the most accessible. Since McFadden's list in 1983, there have been nine new Earth-approaching

asteroids discovered, and two or three more are being found each year.

Accessibility

The accessibility of an asteroid is the quantitative assessment of the ease by which an asteroid may be approached or explored. Ease is quantified by a combination of distance from the Earth, orbital orientation, orbital position and velocity, and gravitational force. The further an asteroid is from the Earth, the greater the energy and time required to reach it. An orbit that is at a highly-inclined angle to the Earth's orbit may be several times more difficult to reach than an orbit that is in the same plane. The relative positions and velocity differences between the two orbits determine when, and how often, travel is practical from a body in one to a body in the other. The gravitational force of an asteroid determines the amount of energy that must be expended to arrive and depart from the surface of the asteroid. In summary, many factors must be considered to determine the accessibility of an asteroid.

Because of the complexity of the accessibility calculations, only the most promising of intercept opportunities have been analyzed in detail, and then only for a period of about 20 years. Detailed analyses has been done for some of the recently-discovered Earth-approaching asteroids. These analyses need to be extended far enough into the future to detect any unusually favorable opportunities. This time period should include any opportunity, or situation (such as possible defensive action or an ambitious rendezvous mission) for which planning would have to be started in the next several years. Typically, three (occasionally as many as eight) new Earth-approaching asteroids are discovered each year. These discoveries regularly provide new candidates for these analyses.

Methodology

The accessibility of selected asteroids was determined by using an assimilation of classical orbital mechanics and recently-developed mathematical techniques. Because of the complexity of the calculations, computer programming was used extensively throughout this project. Calculations and computer programs were validated by comparing the results to known results. No simulated or arbitrary data was used. All input data was real and the most recent that is available. All output was calculated as accurately as practical for the techniques used. Details of computer accuracy are listed in the section on computer facilities.

There are numerous techniques for the calculation of rendezvous trajectories and the accessibility of asteroids. Both closed-form equations and iterative methods are available. Most of these approaches provide the accuracy but not the speed to handle the large number of calculations required to determine accessibility. Main belt asteroids (all those in similar near-circular orbits between Mars and Jupiter) can be assessed with a few typical calculations that will not vary significantly from one to the other. However, Earth-approaching asteroids each present unique opportunities with highly-varying accessibilities. More efficient calculations are needed.

Two new techniques were announced by Ross and Hulkower in 1981, and were demonstrated in the calculation of the accessibility of asteroid Anteros (17:1-5). The first method determined the optimal rendezvous trajectories and optimal time of flight for each relative position of the Earth and the asteroid. The second technique used a plot of these optimal trajectories for all combinations of positions. These plots were

called Prime Rib curves because of the contour patterns produced. A further routine was developed to increase the accuracy and efficiency of finding the best of all of these trajectories. In 1983, Ross and Hulkower made a more detailed analysis of missions (including return missions) to Anteros. Later in 1983, an asteroid which had been discovered in February 1982 (named 1982 DB), was reported to have replaced Anteros as the most accessible known asteroid (11:42-46). For Anteros and 1982 DB, the published analyses list a total of 36 rendezvous trajectories, and show representative Prime Rib curves against which some of the calculations in this paper are compared.

Initially, the Ross and Hulkower technique appeared to be the most promising. However, extensive analysis of the fundamental equations and hundreds of runs of computer programs revealed a number of problems with this approach. The primary problem was a consistent reluctance of the iterative equations to converge. Techniques to enable convergence for one set of data proved to be unstable for new data. When the equations converged, reasonable results were obtained, but when the equations did not converge, no results were obtained. Because the overall analysis depended on 1296 independent runs of the algorithms to produce one plot of accessibility for one asteroid, it was vital that the computer be able to run automatically with no concern about equations converging. It was then learned from Hulkower that he too had trouble with these equations. In a recent paper by Hulkower, Lau and Bender (12:458-461), he abandoned the original approach, and offered a completely new algorithm. The Prime Rib curves were retained as the best presentation of the optimal trajectories.

The calculations of the accessibility of asteroids are based on the Hulkower, Lau and Bender techniques. Orbits of the Earth and selected asteroids were calculated by computer for the period 5 January 1985 to 5 January 2020. These orbits provide the framework in which to apply the analytical techniques. Primary attention was to be given to rendezvous missions, but two-way and fly-by missions were also addressed. Particularly interesting trends beyond 1 January 2010 were sought. The output of optimal trajectories and launch opportunities was compared with results calculated by Ross and Hulkower, Hulkower, Lau and Bender, and with other sources (18:1-16). Calculations for asteroids, scenarios and times not available from other sources are presented in the appendices of this paper.

The following steps summarize the development of the algorithms:

1. A PASCAL computer program was developed to calculate the orbits of the Earth and the asteroids from 05 January 1985 to 02 January 2020.

2. The orbital program was validated against ephemeris data from Astronomical Almanac for the Year 1985 and Almanac for Computers 1985.

3. The orbital program was extended to produce positions and velocities of the Earth and the asteroids in a form usable by the subsequent optimal trajectory algorithms.

4. Initially, the analytical techniques of Hulkower and Ross were converted to PASCAL subroutines.

5. Orbital parameters of the asteroid Anteros were used to exercise the subroutines.

6. Prime Rib curves for Anteros were compared to results published by Hulkower and Ross.

7. The Hulkower and Ross algorithms were abandoned in favor of the new ones developed by Hulkower, Lau and Bender.

8. New PASCAL computer subroutines were written. The original orbital programs and the Prime Rib plotting routines were reusable.

9. The new algorithms were shown to be stable, accurate and efficient.

10. A PASCAL program was developed to associate Julian Dates and true anomalies of the Earth and the asteroids to 2 January 2020. This facilitated finding optimal launch opportunities from Prime Rib plots.

The Ross and Hulkower Technique

As an extension to Battin's work on optimal one impulse transfers between circular orbits, Ross and Hulkower developed a technique to solve one and two impulse transfers between orbits of arbitrary eccentricity (17:1-5). For ease of reference, this technique will be referred to as the Ross technique. Ross used the same geometric representation as Battin and others as depicted in Figure 1.

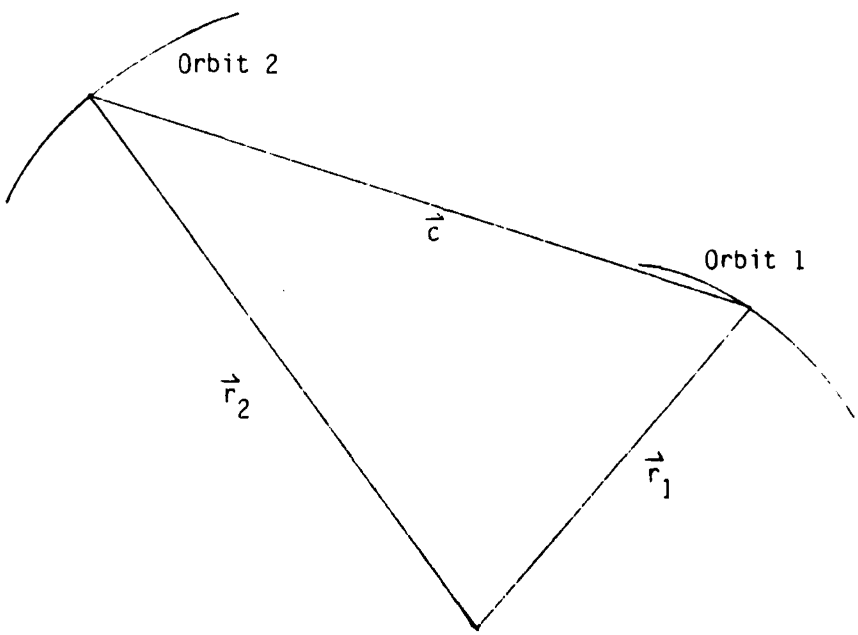


Figure 1. Geometry of the Rendezvous Problem

Position vectors r_1 and r_2 locate the departure planet and arrival planet in relation to the sun. These vectors are joined by a vector c representing the vector subtraction of r_1 from r_2. The angle theta is measured from r_1 to r_2, positive in the direction of rotation of r_1, and ranging from 0 to 360 degrees. Ross establishes three unit vectors for r_1, r_2 and c, and then handles all his equations in a new reference frame defined by sums and differences of these three unit vectors. Departing slightly from his notation, the unit vectors are labelled U_{r_1}, U_{r_2} and U_c. Positions and velocities are then calculated as multiples of the vectors $(U_c + U_{r_1})$, $(U_c - U_{r_1})$ or $(U_c - U_{r_2})$, $(U_c + U_{r_2})$. Although this transformation simplifies the equations used to solve the optimal impulse problem, it considerably increases the manipulation required to handle these equations on a computer. The following equations necessary to *implement the computer solution are* extracted from the much more detailed development by Ross (17:2-3), and Battin (2:93-112):

$$\vec{V}_1 = K[q \sec\varepsilon (U_c + U_{r_1}) + SG \tan\varepsilon (U_c - U_{r_1})] \qquad (1)$$

$$\vec{V}_* = \alpha_*(U_c + U_{r_1}) + \beta_*(U_c - U_{r_1}) + \gamma_* \hat{h} \qquad (2)$$

$$\vec{V}_2 = -K[q \sec\varepsilon(U_c - U_{r_2}) + SG \tan\varepsilon(U_c + U_{r_2})] \qquad (3)$$

$$\vec{V}_p = \alpha_p(U_c - U_{r_2}) + \beta_*(U_c + U_{r_2}) + \gamma_p \hat{h} \qquad (4)$$

$$k = \sqrt{\mu c/[2s(s-c)]} \qquad (5)$$

$$\hat{h} = q \frac{\vec{r}_1 \times \vec{r}_2}{r_1 \, r_2 \sin \theta} \qquad (6)$$

$$s = \frac{r_1 + r_2 + c}{2} \qquad (7)$$

$$q = \text{sign of } (\vec{r}_1 \times \vec{r}_2) \cdot \vec{h}_* \qquad (8)$$

$$k_1 = \frac{1}{\sqrt{|\Delta\vec{V}_1|^2 + \frac{2\mu_*}{r_*}}} \qquad (9)$$

$$k_2 = \frac{1}{\sqrt{|\Delta\vec{V}_2| + \frac{2\mu_p}{r_p}}} \qquad (10)$$

where

$\vec{V}_1$ = transfer velocity departing orbit 1

$\vec{V}_2$ = transfer velocity arriving at orbit 2

$\vec{V}_*$ = departure body's velocity

$\vec{V}_p$ = arrival body's velocity

α,β,γ = velocity components in new reference frame

U_c, U_{r_1}, U_{r_2} = unit vectors as described in text

k = defined quantity

$\vec{r}_1, \vec{r}_2, \vec{c}$ = position vectors from Figure 1

r_1, r_2, c = magnitudes of position vectors

$\hat{h}$ = angular momentum in new reference frame

q = defined as plus or minus

k_1, k_2 = defined quantities

s = defined quantity

θ = angle between $\vec{r}_1$ and $\vec{r}_2$

SG = +1 for transfer orbits not passing through apoapsis between $\bar{r}_1$ and $\bar{r}_2$

SG = -1 for transfer orbits passing through apoapsis between $\vec{r}_1$ and $\vec{r}_2$

ε = free variable in iterations

$$\tan \varepsilon = \frac{q}{2k(k_1-k_2)} \left[k_1 \alpha_* \left(1 + \frac{r_2 \cos \theta}{c} - \frac{r_1}{c}\right) \right.$$

$$\left. + k_2 \alpha_p \left(1 + \frac{r_1 \cos \theta}{c} - \frac{r_2}{c}\right) \right] \sin \varepsilon$$

$$+ \frac{SG}{2k(k_1-k_2)} \left[k_1 \beta_* \left(1 - \frac{r_2 \cos \theta}{c} + \frac{r_1}{c}\right) \right.$$

$$\left. + k_2 \beta_p \left(1 - \frac{r_1 \cos \theta}{c} + \frac{r_2}{c}\right) \right] \tag{11}$$

The final equation (11) was derived from previous Ross equations but differs significantly from his version of the same result. It appears that there is at least a typographical error, and probably a development error in the final Ross equation. The differences include the quantity 2k and several plus and minus signs. His equation does not collapse back to his previous result for the one-impulse case as it should by equating k_2 to zero. As well, his equation would not converge to produce an optimal impulse answer. When it was learned that a number of problems had plagued this algorithm, and that a new one had been developed by one of the original authors, this approach was abandoned.

The Hulkower, Lau and Bender Technique

Hulkower, Lau and Bender developed a patched conic iterative method of determining the optimum two-impulse transfers between arbitrary elliptical orbits (12:458-461). For ease of reference, this technique will be referred to as the Hulkower technique. Although this technique is not as elegant as the Ross technique, it is somewhat simpler to implement on a computer. All positions and velocities are calculated in heliocentric-ecliptic coordinates thus requiring no transformations into unusual frames of reference. As with the Ross method, the transfer begins in a circular low Earth orbit and ends in an elliptical or circular orbit around the asteroid. Variations of this basic transfer are allowed for fly-by and for return missions.

Hulkower begins by defining the changes of velocity at the departure and arrival points in terms of the hyperbolic excess velocity, the gravitational constants of the departure and arrival planets, and the periapse and apoapse distances of the departure and arrival parking orbits.

$$\Delta\vec{V}_1 = [\vec{T}_1^2 + (2\mu_1/q_1)]^{\frac{1}{2}} - \{2\mu_1 Q_1/[q_1(Q_1 + q_1)]\}^{\frac{1}{2}} \qquad (12)$$

$$\Delta\vec{V}_2 = [\vec{T}_2^2 + (2\mu_2/q_2)]^{\frac{1}{2}} - \{2\mu_2 Q_2/[q_2(Q_2 + q_2)]\}^{\frac{1}{2}} \qquad (13)$$

where

ΔV = velocity change

$\vec{T}$ = hyperbolic excess velocity

μ = gravitational constants of planets

q = perigee distance

Q = apogee distance

Hulkower adds the two changes in velocity to produce one expression for the total change of velocity. He then takes the partial derivative of this expression with respect to the parameter p of the transfer orbit, producing an expression that equals zero at the minimum total change of velocity. Although this expression is not difficult to handle in the computer program, it was not used in favor of an iterative form of the derivative. However, it can be used to confirm that the optimal trajectory has been found. Ross does not detail his method of finding the optimal trajectories but it probably used this expression.

Following previous work by McCue and Bender (12:459), Hulkower writes an expression relating velocities on the transfer orbit to the hyperbolic excess velocity, the velocity of the departure or arrival planet, unit position vectors, and a defined velocity v. Separate equations are necessary, depending on the size of the angle between the position vectors of the departure and arrival bodies. Following Hulkower, the upper sign refers to angles less than 180 degrees, and the lower sign refers to angles between 180 and 360 degrees.

$$\vec{V}_1 = -\vec{I}_1 \pm (\vec{V} + z\,\vec{U}_1) \tag{14}$$

$$\vec{V}_2 = I_2 \pm (\vec{V} + z\,\vec{U}_2) \tag{15}$$

This defined velocity is expressed in terms of the gravitational constant of the sun, the parameter of the transfer orbit, and the position vectors of the departure and arrival planets in the heliocentric-ecliptic reference frame.

$$\vec{V} = [(\mu p)^{\frac{1}{2}} (\vec{r}_2 - \vec{r}_1)/ |\vec{r}_1 \times \vec{r}_2| \tag{16}$$

The angle between the position vectors is used to calculate a quantity z.

$$z = (\mu/p)^{\frac{1}{2}} \tan(\Phi/2) \tag{17}$$

where

θ = angle between $\vec{r}_1$ and $\vec{r}_2$

The free variable that is used to find the optimal transfer trajectory is the parameter of the transfer orbit. Hulkower gives expressions for the upper and lower bounds of the parameter to simplify the search process. These maximum and minimum values of the parameter are given in terms of the position vectors of the departure and arrival points.

$$p_{min} = \frac{r_1 r_2 - \vec{r}_1 \cdot \vec{r}_2}{r_1 + r_2 + [2(r_1 r_2 + \vec{r}_1 \cdot \vec{r}_2)]^{\frac{1}{2}}} \tag{18}$$

$$p_{max} = \frac{\vec{r}_1 \vec{r}_2 - \vec{r}_1 \cdot \vec{r}_2}{r_1 + r_2 - [2(r_1 r_2 + \vec{r}_1 \cdot \vec{r}_2)]} \tag{19}$$

Hulkower plots the optimal changes in velocity using the same Prime Rib curves that were used for the Ross method. Each point shows the total change of velocity for a given combination of true anomaly of the departure planet at launch and the true anomaly of the arrival planet at arrival. Figure 2 shows the essential elements of a Prime Rib curve. Figure 3 shows an example of the Prime Rib plot generated by the program in this paper. This particular plot is for the asteroid 1982 DB, the most accessible asteroid yet discovered.

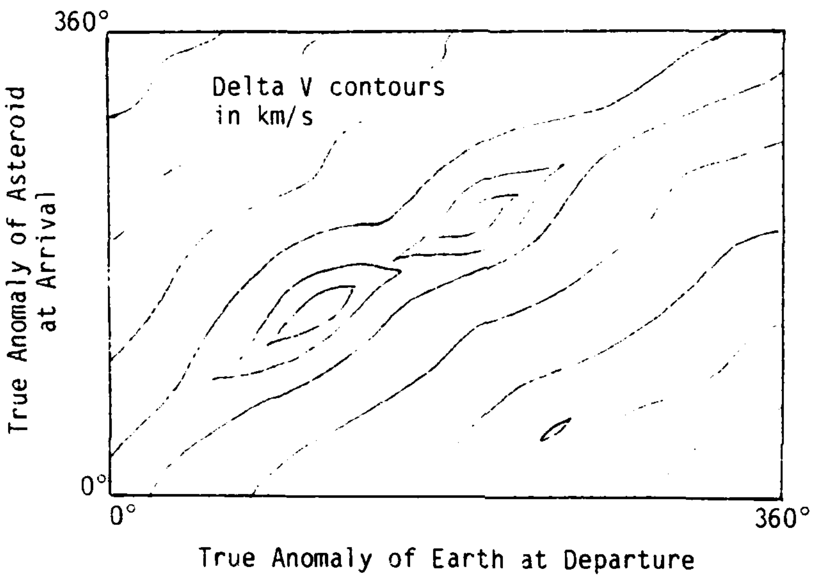
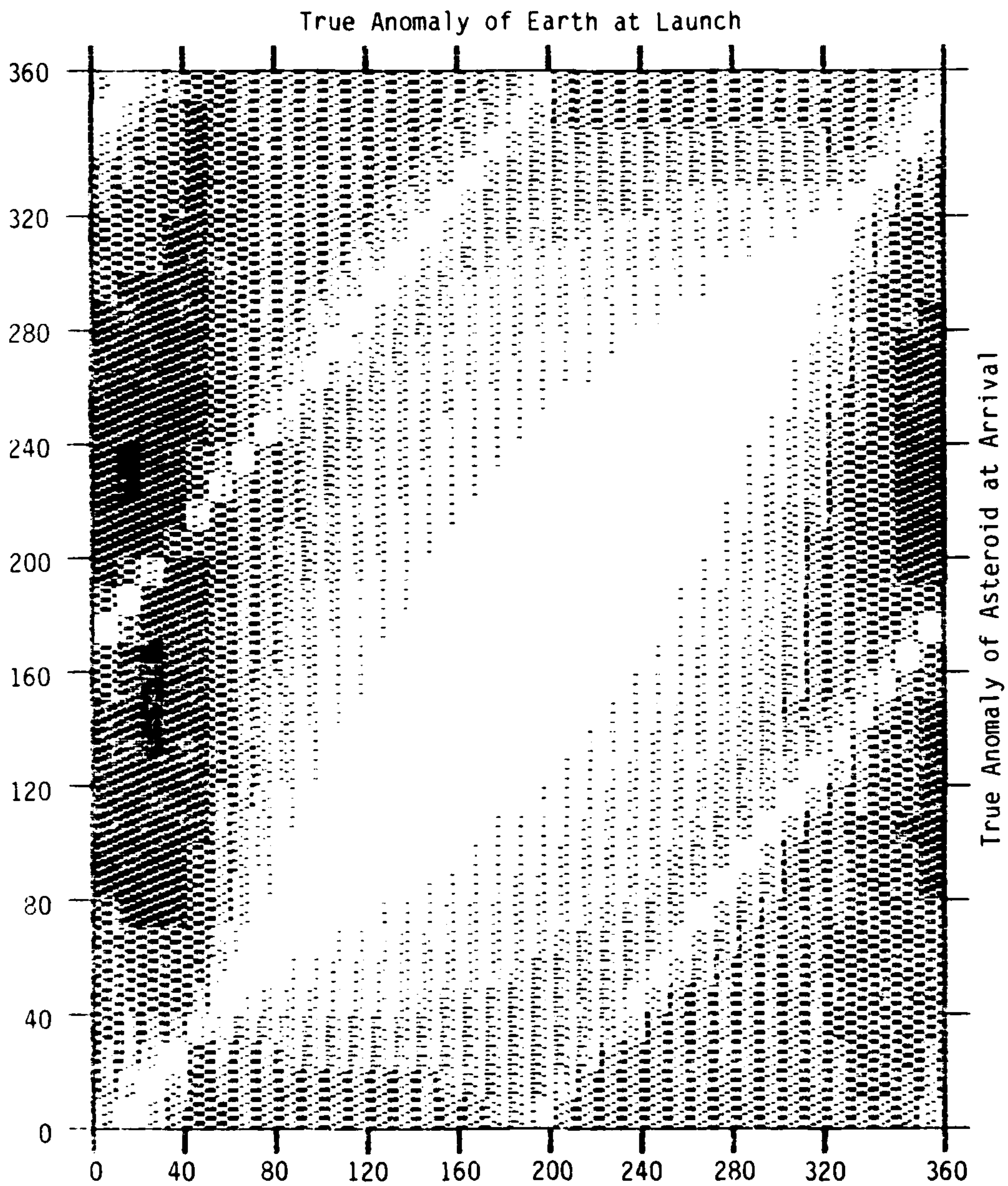


Figure 2. Essential Elements of a Prime Rib Curve

ASTEROID 1982DB



(Grey Scale: Black ≤ 4.5 km/s. White > 20 km/s)

Figure 3. Example of a Prime Rib Plot

V. Computer Program Development

Common PASCAL Procedures

GetDate. This procedure is used to obtain both the start and the end date, and the time of an ephemeris period. The input must be properly entered or the procedure will reject it and ask again. The month must contain three letters, with the first letter upper case and the remaining two letters lower case. The day of the month is error checked and must be an integer between one and the maximum days for that month. The year must be an integer between 1900 and 2099, the limits of the Julian date algorithm used in the program (20:B2). The hour of the day is the Coordinated Universal Time (UTC) input as a real number between 0.0 and 24.0. The procedure returns the year, month, day and hour as real numbers.

CalendarDate. This procedure accepts a Julian date and returns a real number representation of the day, month and year. Leap years are handled correctly.

JulianDate. This procedure is the complement to CalendarDate. It accepts a year, month, day and decimal hour (UTC) and returns a real number Julian Date. The full seven digits plus the decimal part of the Julian date are returned. The algorithm is from the Almanac for Computers 1985 (20:B2).

SolveKeplersEqn. This procedure accepts a mean anomaly in degrees and returns the eccentric anomaly in degrees. It is implemented using a traditional iterative approach, with a selectable accuracy.

SunXYZCoord. This procedure accepts the Julian date and uses the global variables of the Earth's orbital elements in the heliocentric-ecliptic coordinate system. Outputs are the X,Y,Z coordinates of the sun in the geocentric-equatorial coordinate system. The obliquity of the ecliptic is a variable adjusted from a reference date. This algorithm was adapted from the low-precision formulae for the Sun's coordinates from the Astronomical Almanac 1985 (20:C24). The stated precision is 0.01 degree between the years 1950 and 2050. This formula sets the limits over which this program can be used. Greater accuracy would require either vastly more complex formulae, or a series expansion technique such as that used to generate ephemerides in the Astronomical Almanac. The latter is very accurate but can only be used over the period for which the coefficients for the expansion are tabulated in the almanac (usually one year).

GetDataFile. This procedure handles the data file manipulation for most of the programs used in this paper. The data files are stored as text files so they can easily be viewed, corrected or updated. All the data files are formatted in the same way. They contain the name and/or number of the solar system object plus the classical orbital element set referenced to the heliocentric-ecliptic coordinate system. Although the programs which use this procedure will normally be concerned with the Earth and Earth-approaching asteroids, this procedure was written to accept a variety of data files. All nine planets are included in the file PLANET.ELE, which contains the 1985 orbital elements. Rendezvous missions could be calculated for Mars and Venus to compare the trajectories and the energy requirements with those for missions to the

asteroids. Several well-known comets are included in a file called COMET.ELE. The main purpose of this file was to verify the ephemesis program by comparing the output to published ephemerides of Halley's comet. The similarity of some cometary orbits to those of asteroids permit comparisons of rendezvous trajectories. Several files are available for the low-numbered asteroids, again because these objects are well observed and ephemerides are available. Specific comparisons were made of published and generated ephemerides for asteroids 1 Ceres, 2 Pallas, 3 Juno and 4 Vesta.

The input menu for this procedure suggests using one of the standard files for planets, comets or asteroids, and allows an optional user defined file to be used. For some programs used in this paper, the user defined option was replaced by a specific file of AAA asteroids.

InitialDisplay. This procedure generates the initial display for the main program. It introduces the program and lists which data files are available on the data disk. It also controls the input of the name of the optional user-supplied data file if that option was selected. No error checking is done to see if a user-supplied data file exists before access is attempted, so the user must ensure that the proper data file is on the default disk drive.

PASCAL Program HelioCentricOrbit

The program HelioCentricOrbit is used to generate ephemerides for solar system objects (1:51-83; 7:90-98). The program computes right ascension and declination using classical Keplerian motion. No perturbation corrections are incorporated. Most of the common PASCAL procedures listed in the previous section are used to initiate the program

and assemble the data files. Any of the data files can be used, permitting the ephemerides of planets, comets and asteroids to be generated. The period of validity of the program is 1950 to 2050. The accuracy depends on the algorithm for the position of the sun discussed in the procedure SunXYZCoord, and the effects of the perturbations.

CalculateObjPosition. This procedure is the heart of the program. It first determines the position of the sun in the geocentric-equatorial reference frame by calling the procedure SunXYZCoord. The classical element set is then used to calculate the position of the solar system object in the heliocentric-ecliptic reference frame. Traditional methods are used in these calculations, such as the use of auxiliary quantities and Gaussian constants. The procedure to solve Kepler's equation is called to calculate the eccentric anomaly. All positions are transformed into the geocentric-equatorial frame and the right ascension and declination are generated from trigometric calculations.

PeriodEphemeris. This procedure permits the user to establish the period for which the ephemeris is to be calculated, or optionally, a single date and time. If a period of time is selected, the user is asked for the start and end dates and times, and the interval of time between each calculation. All the dates are converted to seven digit Julian dates for ease of calculations. Output may be directed to either the screen or printer. The same format is output to whichever display is selected. The output consists of the object name, the day, month and year, and the right ascension and declination in hours and degrees. This procedure uses two internal procedures called CalcEphemeris and Print to control the calculations and to generate the output.

PASCAL Program RossHulkowerDeltaVee

This program uses the Ross and Hulkower algorithms to generate a plot of optimal trajectory change in velocity or delta vee. For simplicity, this technique will be referred to as the Ross technique. Parts of the program are derived from the program HelioCentricOrbit. This program is self contained, except for the data files. It generates positions and velocities for both the departure and arrival planets and runs the optimal trajectory algorithms. Output can be either a Prime Rib plot of delta vee for each combination of true anomalies of the departure and arrival planets, or a tabular listing of the same information.

Several functions such as "tan" or tangent are programmed in this program because they are not part of the fundamental PASCAL compiler. These functions are derived from the sine, cosine and arctangent functions using standard trigometric relations. No further explanations will be given for these and similar functions as their implementation is self-evident in the program listing.

Because the procedures and algorithms in this program must be repeated 1296 times to generate one Prime Rib plot, efficiency is an important consideration. Several of the procedures from the program HelioCentricOrbit have been modified to increase their efficiency. Quantities that have to be computed only once and remain constant for a number of iterations are often broken out into separate procedures. Modifications of the previous procedures to accomplish this efficiency are obvious in the program listings and will not be discussed further.

CalculateObjPosition. This procedure is identical to the procedure of the same name in the program HelioCentricOrbit.

CalculateObjVelocity. This procedure calculates the vector velocities of the departure and arrival planets. These vector quantities are computed from the positions, orbital elements and heliocentric gravitational constant using standard formulae (13:300).

Departure360. This procedure uses the previous two procedures to calculate the position and velocity vectors for the departure planet every ten degrees of true anomaly from 0 to 360 degrees. The ten degree increment is arbitrary and is selected only because it is convenient. This selection provides a compromise between speed of execution and resolution of output. Any increment could be substituted without affecting the proper functioning of the program, but a smaller quantity (such as one degree) could greatly increase the run time of the program. If the output is to the Prime Rib plot, provision must be made for the different resolution. A practical compromise would be to run the program first with ten degree resolution to determine areas of interest, and rerun the program in higher resolution over a limited combination of true anomalies. The output of this procedure is stored in two 2-dimensional arrays of position and velocity for the departure planet.

Arrival360. This procedure functions exactly like the previous one. The reason for having a separate procedure for the departure and arrival planets is to facilitate operating the program over a limited range of true anomalies that probably will be different for each planet.

GetAlphaBetaGamma. This procedure converts the position and velocity data from the heliocentric-ecliptic reference frame into a skewed frame used by the Ross algorithms. The Alpha, Beta and Gamma are used to be consistent with the Ross notation, and refer to the three components of

velocity. In the arrays, [1], [2], and [3] correspond to Alpha, Beta and Gamma.

The transformations are computed by the dot product of the velocity vectors and a unit vector in the new frame as established by Ross. Once the transformation has been completed, the calculations are predominantly 2-dimensional in the plane defined by the transfer orbit. Three-dimensions are used when the results are changed back to the heliocentric-ecliptic frame. The angular momentum of the departure planet and the cross product of the two position vectors are used to establish the positive orientation for the orbit direction.

<u>UnitVectorcr1r2</u>. This procedure is key to the transformation required in procedure GetAlphaBetaGamma. It provides the unit vectors that define the new reference frame established by Ross (17:2). This new basis is orthogonal, but the values of Alpha, Beta and Gamma are not direct transormations to these unit vectors. Rather, they are referred to the sum and difference of pairs of these unit vectors. This causes the quantities Alpha, Beta and Gamma to be distorted versions of the true velocity vectors. In other words, the magnitude of the velocity after the transformation is not the same as the magnitude of the velocity before the transformation. This must be taken into consideration when these quantities are converted back to the original reference frame.

<u>GetNextSetTrueAnomalies</u>. This procedure accesses the arrays containing the positions and velocities for the departure and arrival planets in preparation for calculating the Alpha, Beta and Gamma program, and accessed as needed by the delta vee part of the program. This

permits more flexibility in limiting the combinations of true anomalies over which the delta vee algorithms will run. As well, it permits the program to be more modular in design to facilitate modifications.

FindDeltaV. This subroutine is the heart of the Ross algorithm to find the optimal trajectories. All the previous procedures were merely setting the stage and assembling the data for this procedure. All variables required by this procedure are passed to it by the calling statement. No global variables are accessed from within the procedure. This permits FindDeltaV to operate as a unit independent from the rest of the program. As such, it could be easily transported to other programs. The main part of this procedure begins by defining several constants according to Ross. The entire procedure is a double iteration. The solution ultimately requires that the quantities DeltaV1 and DeltaV2 be found. These quantities are used to calculate k_1 and k_2 which are in turn used to calculate epsilon. Epsilon is then used to calculate DeltaV1 and DeltaV2, and the cycle repeats until the desired accuracy is reached. The equations for k_1 and k_2 incorporate the size of the parking orbit and the gravitational constant for both the departure and arrival planets. This permits these orbits to be arbitrarily chosen. As well, by equating k_2 to zero, no orbit is established at the arrival planet, and the result is a fly-by mission.

FindEpsilon. This procedure is used to calculate epsilon for a given k_1 and k_2. Because epsilon cannot be isolated on one side of the equation, an iterative solution is required. The iteration is very sensitive to initial conditions, and can diverge and produce no solution. When it does converge, it does so very rapidly. Some of the problem

with the converging can probably be attributed to the differences in the Ross formula for epsilon and the one derived in this paper. The Ross formula appears to have at least typographical errors. The quantity (2k) is missing from the denominators, and several of the plus and minus signs are switched. Deriving this equation from Ross' previous equations produced the equation used in this procedure. Ross' equation for Epsilon would not function at all.

Delta360. This procedure controls the procedures that arrange the data and calculate the delta vee. It effectively separates the first part of the program which calculates the positions and velocities from the second part which implements Ross' algorithms.

PASCAL Program HulkowerLauBenderDeltaVee

This program was developed to replace the RossHulkowerDeltaVee, and was used to generate all the optimal trajectory Prime Rib plots, tabular form of the Prime Rib plots, and the times-of-flight associated with each trajectory. The program uses the Hulkower, Lau and Bender algorithms (12:458-461). For simplicity, Hulkower will be used to mean Hulkower, Lau and Bender. These algorithms were found to be highly-stable and accurate for all combinations of true anomalies, and for all the combinations of departure and arrival planets that were tried. The algorithms presented by Hulkower were changed and simplified for the computer program with no apparent loss of stability or accuracy. These changes are documented in the explanations of the individual procedures. As with the previous programs, use was made of the common procedures for file input and initial calculations to set up the data for the Hulkower algorithms. Some changes were made to these procedures to account for slightly

different data requirements, but these changes are minor, and are readily apparent in the PASCAL code. No additional explanation will be given here.

Diffl. This procedure is the heart of the implementation of Hulkower's algorithms. It uses three internal procedures called FindDiff, Newton and GaussGodalTimeOfFlight which are unique to this program. The procedure accesses the global variables for departure and arrival planet position and velocity vectors, and transfers the data to local variables using the Hulkower notation. Several quantities that remain constant are precomputed prior to calling the other procedures. The parameter of the transfer orbit is used as the free variable that is iterated to find the optimal trajectory. To reduce the number of iterations, and to provide stability to the search routines, upper and lower bounds are computed for the parameter. The starting point for the search routines then becomes the value of the parameter halfway between its computed maximum and minimum value. The Hulkower algorithms identify two cases of the transfer orbit geometry that must be handled differently. The cases are determined by the size of the angle between the position vectors of the departure and arrival planets. If the angle is less than or equal to 180 degrees, then one set of equations apply. If the angle is greater than 180 degrees, then a second set of equations apply. Fortunately, the only difference between the two sets of equations is the plus or minus signs for a number of terms in the equations. For the computer implementation, a quantity call SL is set to either +1.0 or -1.0 and is inserted throughout the equation. The optimal trajectory is computed for the appropriate case. This feature of the algorithm provides a check on its proper

functioning. If both cases are run through the optimal trajectory procedure, the case which gives the minimum delta vee should be the one that is appropriate to the size of the angle between the position vectors of the departure and arrival planets. This check for the proper functioning of the algorithms was retained in the program that is listed in Appendix A. However, it is not essential to the operation of the program and could be removed to provide an increase in speed of the program.

Newton. This procedure is called when the preparatory calculations described in the previous procedure have been completed. The purpose of Newton is to iterate the parameter of the transfer orbit, beginning at a point half way between the calculated extremes and remaining at all times between these extremes. For a given value of the parameter, the procedure FindDiff is called to find the difference between the delta vee for that value of the parameter and the delta vee for the previous iteration of the parameter. The value of the parameter for the next iteration is determined by halving the previous iteration step and moving either plus or minus depending on the result of the previous two iterations. When the difference in the total delta vee converges to a defined accuracy, the iteration terminates and the delta vee for that value of the parameter is noted. To accommodate the situation where the iteration oscillates between two very small values, a limit of 50 iterations is maintained. Convergence normally occurs after about 30 iterations, and is not very sensitive in accuracy. Near the optimal point of the curve of this function, the slope is very flat and gradual, and high accuracies are easy to attain.

GaussGodalTimeOfFlight. This procedure is designed to take the few bits of information about the transfer orbit that have so far been found and calculate the time of flight of the transfer trajectory. Gauss and Godal each developed formulae to solve the time of flight problem (2:82-84). However, each solution depended on knowing information about the orbit that is not available from the Hulkower algorithms. By combining parts of the Gauss and Godal solutions, an equation was found that included only known terms. Although this equation is a large and awkward one, it is effective and stable for all orbits investigated in this paper. Care must be taken with the computer implementation of this equation to prevent overflow and excessive round-off errors because it contains several cubes and fourth-roots of transcendental expressions. Inputs to the procedure are four scaler quantities (the magnitude of the two position vectors of the departure and arrival planets, the angle between these vectors, and the parameter of the transfer orbit). The output is the time-of-flight. With the time-of-flight known, all the other elements of the transfer orbit can be calculated if required. Therefore, this procedure provides the important link between delta vee plots of the Hulkower algorithms and complete knowledge of the transfer orbit.

PASCAL Program CalcAnomalies

This program is designed to calculate the true anomalies of both the departure and arrival planets every ten days from 05 January 1985 to 02 January 2020. It generates a data file called ANOM-1.DTA that can be accessed by the program ListAnomalies. This program does not incorporate a procedure to access the data files containing the orbital elements of all the selected solar system objects. The element set for the Earth is

included but the element set for the target must be provided. The program was designed like this so it could readily be used as a procedure within an all-encompassing program. Besides generating the data file, this program simultaneously lists the true anomalies in the same format as the program ListAnomalies. The reason that two separate programs were written is that the data files can be accessed much faster than the data can be calculated. It is therefore prudent to calculate this data once and store the results. This is particularly important when the data must be accessed numerous times by various search routines.

KeplerEquation. This procedure solves the classic Keplerian equation for the eccentric anomaly given the mean anomaly. The algorithm was adapted from one presented by Danby and Burkardt (5:95-107) that features quintic convergence. It is extremely fast, achieving convergence to at least ten decimal places in three iterations or less in almost all cases. It continues to perform at this level even at eccentricities greater than 0.9999. This speed ensures that this procedure and program operate efficiently in calculating large numbers of true anomalies.

Bacon. This procedure calculates the true anomalies from the element set using the procedure KeplerEquation.

SaveAnomalyFile. This procedure generates the data file ANOM-1.DTA from the two arrays of true anomalies. The data file is formatted to be compatible with the program ListAnomalies.

PASCAL Program ListAnomalies

This program is designed to read a data file ANOM-1.DTA that was generated by the program CalcAnomalies. It will list true anomalies for

both the departure and arrival planet every ten days from 05 January 1985 to 02 January 2020. The data is loaded into two 1-dimensional arrays that each hold 1278 entries. If a higher resolution of days was required (such as every two days), the array size and the maximum number of anomalies would have to be adjusted to match those established by the program CalcAnomalies. The output is formatted with Julian dates in increments of 100 days in the left column, and in increments of 10 days across the top.

PASCAL Program FindLaunchDate

This program finds a launch date (if one exists) given a true anomaly of the departure planet at launch, the true anomaly of the arrival planet at arrival, and the time-of-flight. It loads a data file created by the program CalcAnomalies containing the true anomalies of the departure planet and the arrival planet every ten days from January 1985 to January 2020. The search routine asks for the two true anomalies and the time-of-flight. It then expands the search to include the original true anomalies plus and minus ten degrees, and the original time-of-flight plus and minus ten days. All combinations of three values of three quantities are included. Thus, for each set of inputs, 27 searches are conducted to include all adjacent values. This expansion is done to account for the resolution of ten degrees and ten days with the associated round-off errors. It also permits an area search rather than a point search. The validity of an area search is readily seen from the Prime Rib plot where it is apparent that adjacent values of delta vees are usually quite similar for values of true anomalies that vary ten degrees or more. Output from this program converts Julian

dates to day, month and year (the launch date), and prints the associated true anomalies and time-of-flight found in the search routine.

LoadAnomalyFile. This procedure loads the 1278 sets of true anomalies for ten day increments from January 1985 to January 2020. This data is found in the data file created by the program CalcAnomalies called ANOM-1.DTA or in a user-supplied file of the same format. The true anomalies are stored in the file as real numbers with decimals. LoadAnomalyFile converts these anomalies to integer quantities rounded to the nearest ten degrees. The integer days are also rounded to the nearest ten days.

JtoD and WriteDate. These procedures convert the Julian date derived from the array position and a constant to a calendar date. The result is printed when needed in the form DD Month YYYY (i.e., 21 January 1998).

InputTAandTOF. This procedure asks the user for the true anomalies of the departure and arrival planets (in degrees) and for the time-of-flight in days. This information is derived from the Prime Rib and the time-of-flight tables. Typically, the user would select a combination of true anomalies that represented a minimum value of delta vee on the Prime Rib plot and the program would respond with the dates if any when these combinations of true anomalies and time-of-flight would occur during the period from January 1985 to January 2020.

DoCombos. This procedure and its embedded procedures expand the search area to include three values of each of the three inputs. All 27 combinations are then checked for occurrence during the time period. All matches that are found are printed, and the program asks if further runs are required.

Computer Resources

These programs were designed to operate efficiently on a modest microcomputer system using a widely-available computer language.

The programs were developed on an IBM Personal computer with 640 kilobytes of internal memory. The system was equipped with two floppy disk drives each with a capacity of 360 kilobytes per disk. An 8087 math co-processor integrated circuit chip was installed to increase the speed of mathematical operations. A monochrome monitor displayed 25 lines of 80 characters. The monitor was also used as a monochrome graphics terminal with a resolution of 640x200 pixels. Printout was directed to an Epson FX-80 dot matrix printer. It was also used as a graphics terminal to reproduce the 640x200 plots displayed on the monochrome monitor.

All programming was done in PASCAL using the Borland International Turbo PASCAL compiler/editor system. To take advantage of the math co-processor chip, Version 3.0 of Turbo PASCAL with 8087 support was used for all programs.

A minimum system to support these programs would require 256 kilobytes of internal memory, one disk drive and a monochrome display. The programs are usable (with some inconvenience) without the graphics output, without the speed of the math co-processor chip, and without the printer.

VI. Results and Discussion

The results of this analysis of the accessibility of Earth-approaching asteroids consist of computer programs to implement the algorithms, Prime Rib plots and tables to show the global optimum Delta Vee information for selected asteroids, and tables of launch opportunities for selected asteroids from the year 1985 to 2020.

Listings of the PASCAL code for the computer programs are given in Appendices A, B, C, D, E and F. These listings are important because they represent a very practical tool for anyone with an interest in the accessibility of not only the asteroids analyzed in this paper, but for the analysis of any asteroid, comet or planet in the solar system. The orbital elements of selected solar system objects are given in Appendix M. The volume of output that would be generated to analyze just a few asteroids is very large. Only a selected sample of the more interesting and recently-discovered asteroids can be presented here. Therefore, the computer programs are a must to do a detailed analysis of a particular target of interest. These programs also provide a tool to quickly analyze future discoveries. Since these discoveries have been occurring at the rate of between three and eight per year in recent years, and are likely to continue with the renewed interest in this area, these computer programs will provide a readily-accessible source of detailed study as soon as the initial orbital elements of the discoveries are known.

The Prime Rib plots and tables provide a detailed look at the global picture of the accessibility of selected asteroids in a form that is easy to comprehend. They show not only the specific information for one

asteroid but can be viewed together to quickly appreciate the relative accessibility of a number of asteroids. These Prime Rib plots are somewhat different from the Prime Rib curves presented by Ross and Hulkower in that they employ shading rather than contour lines to convey the information. The shading has the advantage that it is readily apparent which plots or segments of plots portray the best potential opportunities. Very simply, the darker the plot, the lower the required delta vee. The plots with the largest areas of dark shading further indicate the asteroids for which the most launch opportunities are likely to exist. The plots are much easier to generate than the curves and can readily be produced on a dot-matrix printer that is a part of most modest computer systems.

The tables of launch opportunities cover the period from 1985 to 2020 for the three most accessible asteroids. These tables retain the true anomaly increment at 10 degrees while Lau and Hulkower (14:1-27) refined the increment considerably, producing additional launch opportunities. This paper introduces a detailed analysis of three new Earth-approaching asteroids that have been discovered in 1985. Launch opportunities for these new asteroids are also tabulated from 1985 to 2020.

Computer Programs

This section is a summary of the function of computer programs that have been developed to support the analyses presented in this paper. Further details will be apparent in the easily-read PASCAL code in Appendices A through F. Each program is designed to operate interactively with the user and no further explanation should be required to use each effectively. Suggested extensions to the programs are proposed

in the following chapter of this paper. Each program is listed by the descriptive name that is embedded in the PASCAL code.

Program	Function and Brief Description
1. HelioCentricOrbit	This is a validation program that demonstrates the accuracy of the algorithms that calculate the orbital positions of solar system bodies. This program is embedded in both the Delta Vee programs to provide the fundamental position and velocity data required by the Delta Vee algorithms. A sample of the accuracy of this program compared to published Astronomical Almanac ephemerides is tabulated in Appendix L.
2. RossHulkowerDeltaVee	This program implements the algorithms introduced by Ross and Hulkower (17:1-5). As explained previously, these algorithms were found to be unstable and difficult to handle. This program is presented because of the suggestions offered in the next chapter of this paper.
3. HulkowerLauBenderDeltaVee	This program is by far the most important to the analyses discussed in

this paper. It produces the Prime Rib plots and tables that depict the global accessibility of asteroids. It also produces a corresponding time-of-flight for each trajectory shown in the Prime Rib plots. The output from this program is used by the following programs to generate launch opportunities.

The strong points of this program are its stability, its accuracy, its speed and its flexibility. Throughout its many runs on a variety of asteroids and other bodies, it was 100% stable and showed none of the problems of iterations failing to converge that were found in the RossHulkowerDeltaVee program. Comparisons with Hulkower, Lau and Bender data showed an accuracy of about 0.01 kilometers per second for delta vees of between 5.0 and 60.0 kilometers per second, and accuracies of between 1 and 20 days for times-of-flight up to 700 days. Suggested improvements to these accuracies are discussed in the following chapter.

This program operates very quickly considering the massive amounts of iterations required by the algorithms. One Prime Rib plot, a table of delta vees and table of times-of-flight showing 1296 individual rendezvous trajectories are produced in a total time of 15.5 minutes. This time is constant for any asteroid. The flexibility of this program permits it to use any planet, comet or asteroid in orbit around the sun as either departure or arrival body. Rendezvous, fly-by, and return missions are all supported.

4. CalcAnomalies	This program calculates true anomalies for a pair of bodies (usually the Earth and an asteroid) every ten days from 5 January 1985 to 2 January 2020. It operates in less than a minute producing a file that is used by the following programs.
5. ListAnomalies	This program produces a listing of the data generated by CalcAnomalies. It operates at the speed of the printer used for output.

6. FindLaunchDate — This program finds any launch opportunities that exist for a given combination of true anomalies for the period 5 January 1985 to 2 January 2020. The accuracy within this program is plus or minus five days based on one half the increment of the search. Despite having to investigate 27 combinations of true anomalies and times-of-flight for a period of 35 years, it operates with no perceptable delay.

Prime Rib Plots and Tables

Prime Rib plots are displayed for asteroids 1982DB, 1982XB, Anteros, 1985JA, 1985PA and 1985TB in Appendix G. Tabulated values of the plots and times-of-flight are very lengthy. Therefore, only the values for one selected asteroid are shown. The asteroid selected is 1985TB because it was the most recent and the most accessible of the three Earth-approaching asteroids discovered in 1985. The values for 1985TB are detailed in Appendices H and I. The following table summarizes, in order of accessibility, the global optimum trajectory information contained in the detailed tables for all six asteroids. The table also shows the relative accessibility of these asteroids compared to the 76 that have been ranked.

TABLE I

Accessibility of Selected Asteroids

Asteroid	Global Minimum Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)	Rank
1982DB	4.5	20	150	229	1
1982XB	4.2	340	120	185	2
Anteros	5.3	140	240	435	3
1985TB	13.3	320	110	218	61
1985JA	15.7	120	260	407	66
1985PA	21.5	60	220	302	73

One of the common characteristics of all of the Prime Rib plots are the vivid lines of demarcation separating the plots into three areas. Since the plots wrap around both horizontally and vertically, the lines actually separate the plots into two areas separated by 180 degrees. These areas show how the accessibility changes throughout the 360 degree range of true anomalies. If only the true anomalies of the departure planet are considered, one would expect that there would be points on this range where the accessibility is best and points where it is worst. The lines of demarcation are the worst areas and between these lines there are a total of two best points, one usually being significantly better than the other. Like the worst areas, the best areas are separated by about 180 degrees. As the phase of the target true anomaly changes with respect to the departure true anomaly, the lines of demarcation propagate across the plot at an angle that averages 45 degrees. For complex geometries, such as one orbit being highly inclined to the

other, or one orbit being much larger than the other, the plots become more complex with more local minimum and maximum areas.

Prime Rib Plot for 1982DB. This plot shows the global optimal point at about 20 degrees of true anomaly of the Earth and 150 degrees for the asteroid. A second local minimum occurs at 20 degrees and 220 degrees. Although both areas have about the same value of delta vee, the first area is considerably larger than the second. Size of the area has a direct impact on the size of the launch window. The larger the area for a given delta vee, the larger will be the launch window to take advantage of that particular delta vee. Just as important, the larger the area, the higher the probability that the Earth and the asteroid will actually find themselves at that combination of true anomalies. If the area is very small, such as a degree or so across (and beyond the resolution of this particular plot), it will probably be tens or hundreds of years before that particular combination of true anomalies will occur. If the asteroid were in an orbit with a period that was an exact multiple of the period of the Earth, then only a few of the combinations of true anomalies would ever occur. This would mean that parts of the Prime Rib plots, although possibly interesting, would be of no value for mission planning. If the global optimal point occurred in this region, it would never be achievable because the asteroid and the Earth simply would never get into the right positions. For 1982DB, this problem does not exist. The global optimal condition, or a point very close to it occurs frequently. Almost the entire plot for 1982DB shows delta vee values of less than 20 kilometers per second, and large areas less than 7 kilometers per second. These large areas of small delta vees would indicate that

1982DB is a very accessible asteroid. In fact, combined with reasonable times-of-flight and frequent launch windows, these values of delta vees make 1982DB the most accessible asteroid yet discovered.

Prime Rib Plot for 1982XB. This plot is remarkably similar to the one for 1982DB, except that it is displaced slightly along the axis of the true anomaly of the Earth. The global optimal delta vee of about 5.2 kilometers per second makes 1982XB about 0.7 kilometers per second less accessible than 1982DB, but still the second most accessible asteroid.

Prime Rib Plot for Anteros. The plot for Anteros shows global optimal values just slightly worse than 1982XB. Like the other two, Anteros displays large areas of relatively low delta vee enabling the combinations of true anomalies which represent good launch opportunities to occur frequently. Anteros is ranked third in accessibility.

Prime Rib Plot for 1985JA. This plot immediately shows that 1985JA is much less accessible than the three most accessible asteroids. Only the lightest shade of grey was plotted, indicating no values of delta vee below 15 kilometers per second. Most of the plot is white meaning that these values of delta vee are too high to be of any real interest. If these higher values were of interest, the plot could be replotted with the lowest delta vee normalized to black, and the shades of grey extending to higher values. The global optimal point on the plot is 15.7 kilometers per second giving 1985JA a ranking of 66 for accessibility out of 76 that are ranked.

Prime Rib Plot for 1985PA. No values of delta vee plotted for 1985PA with the default setting for shades of grey. The global optimal

point was 21.5 kilometers per second ranking the asteroid 73rd out of 76, just about the most inaccessible of the known Earth-approaching asteroids. This large value is due mainly to the nigh inclination of 55 degrees of the orbit. Plane changes in intercept trajectories are very costly in delta vee. All three of the 1985 asteroids have fairly high inclinations and fairly low accessibility.

Prime Rib Plot and Table for 1985TB. This asteroid is the most accessible of the three discovered in 1985. It shows a plot with two shades of grey, and a global optimal point of 13.3 kilometers per second. That still places it only 61st out of 76 that are ranked. A fairly small area of the plot shows values near 13.3, so launch dates near that value would be infrequent.

Overall, the three 1985 asteroids are very inaccessible compared with the three most accessible. They require about three to five times the delta vee to reach than the most accessible, 1982DB. For comparison, they have roughly the same accessibility as the main belt asteroid Pallas. 1985TB is slightly more accessible than Pallas, 1985JA is about the same, and 1985PA is slightly worse. That comparison demonstrates that asteroids that approach the Earth are not necessarily more accessible than distant asteroids. Inclination is an important factor, as is the shape of the orbit and the velocity of the asteroid when it is near the Earth.

Launch Opportunities to the Three Most Accessible Asteroids

Launch opportunities for the three most accessible asteroids are detailed in the following tables. These opportunities are listed in chronological order from January 1985 until January 2020. They represent

a selection of the optimum and near-optimum trajectories taken from the Prime Rib plots. They are, of course, not the only launch opportunities because given enough delta vee, one can launch any time. Only rendezvous missions are considered here, with the primary consideration being given to delta vee. The time-of-flight could just as easily have been the limiting constraint and would be a very important consideration for a manned flight. A manned flight would naturally include a return trajectory and consideration would be required for how long to remain on the asteroid. A trade-off between delta vee on both legs, launch windows, time-of-flight and time on the asteroid would be a mission planning task. Mission planning is beyond the scope of this analysis. However, the tools presented in these computer programs provide the data that would be required for this kind of mission planning.

TABLE II

Favorable Launch Opportunities
to Asteroid 1982DB from 1985 to 2020

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
3 Feb 1991	4.5	20	150	240
13 Feb 1991	4.6	30	120	150
17 Dec 2001	5.6	330	60	90
26 Jan 2002	4.5	10	120	140
5 Feb 2002	4.5	20	150	220
15 Feb 2002	4.6	20	120	130
7 Mar 2002	5.6	40	180	290
16 Mar 2004	5.8	60	300	490
29 Jan 2011	4.5	20	150	240
29 Jan 2011	4.5	20	230	530
28 Feb 2011	5.6	50	180	310

TABLE III

Favorable Launch Opportunities
to Asteroid 1982XB from 1985 to 2020

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
21 Nov 1987	7.1	310	190	490
1 Dec 1987	6.0	320	80	120
21 Dec 1987	5.3	340	80	110
30 Jan 1988	7.0	20	160	310
24 Dec 1992	5.3	340	100	100
3 Jan 1993	5.4	350	150	300
2 Feb 1993	7.0	20	160	300
8 Dec 1997	6.0	310	70	100
17 Jan 1998	5.7	360	140	230
6 Feb 1998	7.0	20	160	310
22 Nov 2002	7.1	300	190	490
2 Dec 2002	6.0	320	80	100
31 Jan 2003	7.0	20	160	310
26 Nov 2007	7.1	310	180	480
4 Feb 2008	7.0	20	160	300
23 Nov 2017	7.1	310	190	480

TABLE IV

Favorable Launch Opportunities
to Asteroid Anteros from 1985 to 2020

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
4 Jun 1985	5.4	140	270	480
4 Jun 1985	5.6	150	160	200
4 Jul 1987	7.1	170	320	420
1 Jun 1997	5.4	140	270	470
1 Jul 1999	7.1	170	330	420

TABLE IV (continued)

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
5 Jun 2002	5.6	140	150	220
8 Jun 2009	5.4	150	270	460
8 Jul 2011	7.1	170	340	400
2 Jun 2014	5.3	130	250	450
2 Jun 2014	5.6	140	160	220

Launch Opportunities to Earth-Approaching Asteroids Discovered from 1 January 1985 to 13 November 1985

TABLE V

Favorable Launch Opportunities to Asteroid 1985JA from 1985 to 2020

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
17 May 1996	16.5	120	280	430
17 May 1998	15.7	120	260	420
26 Nov 2005	16.8	310	70	200
27 Apr 2017	16.9	100	270	450
17 May 2017	15.7	120	260	410

TABLE VI

Favorable Launch Opportunities
to Asteroid 1985PA from 1985 to 2020

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
30 Jan 1988	22.8	20	230	360
10 Mar 1988	21.5	60	220	300
2 Feb 1993	22.8	20	230	380
14 Mar 1993	21.5	60	220	310
18 Mar 1998	22.5	60	210	310
9 Feb 2005	22.6	30	240	360
31 Mar 2005	22.5	80	220	260
3 Feb 2010	22.8	20	230	370
15 Mar 2010	21.5	60	220	300
19 Mar 2015	22.5	60	210	300

TABLE VII

Favorable Launch Opportunities
to Asteroid 1985TB from 1985 to 2020

Date	Total Delta V (km/s)	True Anomaly of Earth at Launch (deg)	True Anomaly of Asteroid at Arrival (deg)	Time-of-Flight (days)
1 Dec 1985	13.3	320	110	220
20 Jan 1986	14.9	10	120	200
21 Jan 2003	14.9	10	120	190
20 Jan 2005	16.9	10	290	680
19 Jan 2009	16.8	10	280	740
4 Apr 2010	16.9	80	340	380
18 Jan 2013	16.5	0	280	820
17 Jan 2017	16.8	0	270	880

VII. Suggestions and Recommendations

The results presented in the previous chapter are essentially a demonstration of the potential of the programs developed to analyze the accessibility of Earth-approaching asteroids. The analysis process is complete in that it proceeds from fundamental information about the orbits of the asteroids under consideration through the various stages to the launch dates for possible rendezvous missions. This series of computer programs is a first attempt to model the analysis process from start to finish. Many refinements of the programs are readily apparent to make these tools truly useful. Some suggestions for these improvements are presented in this chapter.

Orbit Accuracy

All the orbital motion calculated in these programs has been ideal Keplerian two-body motion. In the short term, for one or two years, this assumption is reasonably valid, especially considering that the initial orbit of newly-discovered asteroids is usually not known very well any way. For asteroids that have been observed on several orbits, and for which the orbital elements have been very accurately determined, increased computational accuracy of the orbital motion may be desirable. This is especially important for planning rendezvous missions to be conducted tens of years in the future. Perturbations in the orbits of the bodies involved can be significant, and of course grow with time. Some orbits are perturbed very little while others could be greatly perturbed, depending on their proximity to the orbits of the larger planets. The

asteroids that most closely approach the Earth are going to be the most perturbed by the Earth.

Considerable effort will be required to modify these programs to account for perturbations. However, since most of the asteroids in question never venture too far from the Earth, a very reasonable approximation could be obtained by considering only the perturbing effects of Jupiter, Mars, Venus and the Earth. But even this simplification still leaves a challenging six-body orbital mechanics problem which would probably have to be solved by numerical methods of computer iterations. This is a slow process if any degree of accuracy is required. Fortunately, the current computer program is not at all time constrained in its calculations of orbits. Almost all of the 15.5 minutes of computer time required by the main program is tied up in the delta vee algorithms. The orbits are calculated completely outside of this time consuming part of the program. There would be no multiplicative effect on time by enhancing the program to account for perturbations.

Delta Vee Algorithms

Most of the time required to solve the delta vee algorithms is consumed in an inefficient routine that finds the minimum value on the curve defined by the two equations that use the parameter of the intercept ellipse as a free variable. About 30 iterations of a bisection search routine are required to converge to a reasonable accuracy. The procedure called Kepler in the program CalcAnomalies was tried as a replacement for the bisection routine. Although the Kepler procedure is extremely fast, it did not often find the global minimum on the curve but converged on a local minimum and was abandoned. However, it is virtually

guaranteed to find a solution in three iterations. If the problem of local convergence could be overcome, the time required for the whole program could be reduced from 15.5 minutes down to about 1.5 minutes. This would offset the inevitable increase in time that would result if the program was modified to account for perturbations.

Integrating All the Computer Programs

Except for the validation program called HelioCentricOrbit which is not essential to the analyses, all the other programs could easily be combined. They were left separated merely as a convenience during program development and testing. A master program, using menu-driven options, would relieve most of the manual entries required by the present programs. All programs rely on a common set of data files containing the orbital elements. There would be less need to generate intermediate results in text files as is now required. The process of combining these programs has been facilitated by ensuring that all of the programming is modular and most of it is portable to other programs.

Launcn Date Search Routines

The program FindLaunchDates does a very efficient search routine for selected combinations of true anomalies and time-of-flight. However, all the input must now be manually entered from the tabulated values of the Prime Rib plots and time-of-flight tables. This is a reasonable approach if only the launch opportunities for a limited number of combinations of true anomalies are required. Unfortunately, this does not ensure that the optimum or even the better launch opportunities for a given period are found. There may indeed be no match found to satisfy

the conditions of the global optimal point on the Prime Rib curves. It then becomes a matter of good luck or laborious manual input to search for the next best solution. There are 1296 possible inputs making an automated global search a very desirable feature. Such a search could be conveniently added when the separate programs are combined into a master program.

Detailed Trajectories

When the optimal delta vees are found for each combination of true anomalies, very little is actually known about the actual trajectory. However, once the parameter and the time-of-flight of the trajectory are calculated, all the other elements of the ellipse fall out of the equations. A pictorial or tabular representation of the rendezvous trajectory could then be constructed to better visualize and understand the geometry of the optimal solutions. This information could reveal fly-by opportunities for other asteroids enroute to the main objective. It could also be used for a check on potential perturbations by observing the path of the trajectory compared to the positions of the major planets.

Return Missions

The current system of computer programs permit return missions to be analyzed but the process requires a considerable amount of manual intervention. The addition of a return mission search routine would greatly enhance the utility of the programs. A further data file would be required containing the gravitational constants of the planets and the larger asteroids. The current programs have this information for only the Earth and a representative value for the small asteroids

assuming a body of one kilometer in diameter. To save on the delta vee requirements for return missions, some are proposed which use atmospheric braking to return to a Shuttle-accessible low Earth orbit. A capability to handle this type of trajectory would be useful for planning return trajectories.

Adjustable Prime Rib Plots

The current procedure that plots the Prime Ribs uses a single fixed grey scale to represent the delta vee values. This is quite useful for comparing several plots but within one plot, it would be better to be able to adjust the grey scale. For example, it might be desirable to always have the optimal area the darkest black, regardless of the actual delta vee that it represents. It might be practical to display only that part of the plot with values of delta vee up to a certain value, such as the maximum delta vee available in a proposed rocket system for a particular mission.

Conclusion

The series of programs that has been presented in this thesis provides a basis of meaningful analyses of the accessibility of Earth-approaching asteroids. These analyses have been conducted for selected asteroids and the results are apparent. However, the potential of these programs is to provide a more accurate and more usable system that can be implemented on modest computer resources to extend the capability of, and access to, these kinds of analyses.

Appendix A

PASCAL Program to Implement the Hulkower-Lau-Bender Algorithms

```
program TrueAnomDeltaVee;

(* Calc Delta V in Space of True Anom + Plot. Obj pos'n, velocity in  *)
(* heliocentric/ecliptic IJK ref frame. Then runs DeltaVee program on *)
(* 360x360 combinations of Departure/Arrival True Anom. Hulkower, Lau,*)
(* Bender Method. Plots Prime Ribs, generates data files for Delta Vee*)
(* and Times-of-Flight.                                               *)

const
   MaxRecSize = 25;
   HelioGravConst = 3.9640157489E-14;   (* AU3/s2  derived from
                                        Astronomical Almanac 85 p. K6 *)
   AU = 1.49597870E8;                   (* km      derived from
                                        Astronomical Almanac 85 p. K6 *)
   GravConstSun = 132718E6;     (* km3/s2 *)
   GravConstEarth = 398603.0;   (* km3/s2 *)

type
    Name = string [10];
    ElementSet = record
          AsteroidName: Name;    (* object name *)
                    Tp: real;    (* date of perih passage or data pt *)
                    i : real;    (* inclination to ecliptic *)
                    w : real;    (* arguement of perihelion *)
                    N : real;    (* longitude of ascending node *)
                    a : real;    (* semi-major axis, in AU  *)
                    e : real;    (* eccentricity *)
                    Mo: real;    (* mean anomoly at To,
                                              Mo = 0 at perihelion *)
                 end;

var
     AsteroidFile: text;
     FileName: string[14];
     ElementSetRec: array [1..MaxRecSize] of ElementSet;
     Ifor: integer;
     MaxNoOfAsteroids: integer;
     Selected: integer;  (* record selected to work with *)

          AsteroidName: name;
                    Tp: real;    (* date of perih passage or data pt *)
                    i : real;    (* inclination to ecliptic *)
                    w : real;    (* arguement of perihelion *)
                    N : real;    (* longitude of ascending node *)
                    a : real;    (* semi-major axis, in AU  *)
                    e : real;    (* eccentricity *)
```

```
                 Mo: real;    (* mean anomoly at To,
                                        Mo = 0 at perihelion *)

    TrueAnomaly, EccentricAnomaly: real;
    Isun, Jsun, Ksun: real;     (* heliocentric/ecliptic coordinates *)
    VelocI, VelocJ, VelocK: real;  (* heliocentric/ecliptic velocity *)
    Rsun: real;                    (* distance from sun *)
    A1,A2,B1,B2,Y1,Y2: real;       (* auxiliaries *)
    PiSun, PjSun, PkSun, QiSun, QjSun, QkSun: real;  (* auxiliaries *)
    sini, sinw, sinN: real;  (* to precompute sin(i) etc.  *)
    cosi, cosw, cosN: real;

    DeparturePosition:     array[0..36,1..3] of real;
    DepartureVelocity:     array[0..36,1..3] of real;
    ArrivalPosition:       array[0..36,1..3] of real;
    ArrivalVelocity:       array[0..36,1..3] of real;
                 U1:       array[1..3] of real;
                 U2:       array[1..3] of real;

    r1,r2:real;

    DepartVel:      array[1..3] of real;  (* old coord sys *)
    ArrivalVel:      array[1..3] of real;  (* old coord sys *)

    DepartPos:  array[1..3] of real;  (* old coord sys *)
    ArrivalPos:    array[1..3] of real;  (* old coord sys *)

    R1xR2: array[1..3] of real;
    MagR1xR2 : real;

    AngleBetweenVectors: real;
    q : integer;
    counter: integer;
    TrueEarth, TrueTarget : integer;

    DataOut : text;
    OutFile : string[8];

    DeltaVeeStore  : array[0..36,0..36] of real;
   TimeOfFltStore  : array[0..36,0..36] of real;


function power(a,b:real):real;
begin
  if (a = 0) and (b = 0) then power := 1
    else begin
           if a>0 then power := exp(b*ln(a))
                  else begin
                         if a<0 then begin
                            writeln(' illegal arg. minus power ');
                            power := -999999999.9;
                                  end
```

```
                                      else power := 0;
                              end;
                    end;
          end;


function tan(x: real) : real;
   begin
     tan := sin(x)/cos(x);
   end; (* tan *)

function arccos(a:real): real;
   var
     temp: real;

   begin
      if a > 1.0 then a := 1.0;
      if a < -1.0 then a := -1.0;
      if a = 0.0 then a := 0.0000000001;     (* avoid divide by zero *)
      temp := arctan(sqrt(1.0 - a * a) / a);
      if temp < 0.0 then arccos := temp + Pi else arccos := temp;
   end;


procedure GetDataFile;  (* read data into global variables *)
begin
  Assign (AsteroidFile,Filename); (* FileName from proc. Init Display *)
  Reset (AsteroidFile);
     Ifor := 0;
     while not eof (AsteroidFile) do
       begin
         Ifor := Ifor + 1;
         with ElementSetRec[Ifor] do
           begin
             readln (AsteroidFile,
               AsteroidName,
                         Tp,
                         i ,
                         w ,
                         N ,
                         a ,
                         e ,
                         Mo);
             end; (* with *)
         end;     (* while *)
  MaxNoOfAsteroids := Ifor;

  writeln('Objects in file');
  writeln(
 'No. Object      Tp         i        w        N       a       e       Mo'
                                                                          );
```

```
for Ifor := 1 to MaxNoOfAsteroids do
  begin
  with ElementSetRec[Ifor] do

    writeln(AsteroidName:10,Tp:13:4,i:9:4,w:9:4,N:9:4,a:8:4,e:7:4,
                                                          Mo:9:4);
  end; (* for *)
  writeln;
end; (* GetDataFile *)


procedure SelectObject; (* Global variables *)
begin
write('Select one ... ');
  readln(Selected);
  writeln;

AsteroidName := ElementSetRec[Selected].AsteroidName;
          Tp := ElementSetRec[Selected].Tp;
          i  := ElementSetRec[Selected].i ;
          w  := ElementSetRec[Selected].w ;
          N  := ElementSetRec[Selected].N ;
          a  := ElementSetRec[Selected].a ;
          e  := ElementSetRec[Selected].e ;
          Mo := ElementSetRec[Selected].Mo;

    writeln(AsteroidName,Tp:10:1,i:8:3,w:8:3,N:8:3,a:8:3,e:8:3,Mo:8:3);
    delay(20);
end; (* SelectObject *)


procedure InitialDisplay;

type
  FileNames = string[14];

var
  DataFileSelection: integer;
  FilesOnDisk: array[1..6] of FileNames;

begin
  FilesOnDisk[1] := 'PLANET.ELE';
  FilesOnDisk[2] := 'COMET.ELE';
  FilesOnDisk[3] := 'AST-0001.ELE';
  FilesOnDisk[4] := 'AST-0021.ELE';
  FilesOnDisk[5] := 'AST-0041.ELE';
  FilesOnDisk[6] := 'ANTEROS.ELE';

  clrscr;
  writeln;
  writeln;
  writeln;
```

```
  writeln('                        HELIOCENTRIC/ELLIPTIC COORDINATES   ');
  writeln('                          for 360 deg. of TRUE ANOMALIES ');
  writeln;
  writeln('                              planets');
  writeln('                                    comets');
  writeln('                                         asteroids');
  writeln;
  writeln('     Data files: contain classic orbit elements.');
  writeln('           Input: Increment for TRUE ANOMALY.');
  writeln('          Output: Heliocentric/Ecliptic Coordinates.');
  writeln;
  writeln('                      ......Data Files on Disk......');
  writeln('                               1. PLANET.ELE');
  writeln('                               2. COMET.ELE');
  writeln('                               3. AST-0001.ELE');
  writeln('                               4. AST-0021.ELE');
  writeln('                               5. AST-0041.ELE');
  writeln('                               6. ANTEROS.ELE');
  writeln('                               7. User-supplied');
  writeln;
  write('      Select ( 1..7 )  ');
    DataFileSelection := 0;
    while not (DataFileSelection in [1..7]) do
      readln(DataFileSelection);
if DataFileSelection < 7 then Filename :=
                                        FilesOnDisk [DataFileSelection]
                              else
                                begin
                                  writeln;
                                  write('Input FILENAME.EXTENTION    ');
                                  readln(Filename);
                                  writeln;
                                end;
end; (* Initial Display *)

procedure ConvertToRadians;
begin
  w := w * Pi / 180.0;
  i := i * Pi / 180.0;
  N := N * Pi / 180.0;
end;

procedure PreComputeTranscendentals;
begin
   sini := sin(i); cosi := cos(i);
   sinw := sin(w); cosw := cos(w);
   sinN := sin(N); cosN := cos(N);
end;
```

```
procedure CalcAuxQuant ;
begin

   A1 := sinN*sinw;   (* auxiliary quantities *)
   B1 := cosN*sinw;
   Y1 := sini*sinw;

   A2 := sinN*cosw;
   B2 := cosN*cosw;
   Y2 := sini*cosw;

                                                 (* GAUSSIAN CONSTANTS *)
   PiSun := B2 - A1*cosi;
   PjSun := (A2 + B1*cosi);
   PkSun := Y1;
   QiSun := -B1 - A2*cosi;
   QjSun := (-A1 + B2*cosi);
   QkSun := Y2;

end;  (* CalcAuxQuant *)


procedure CalculateObjPosition;

(* Isun, etc are in AU's *)

  begin

   EccentricAnomaly := 2.0 * arctan(  tan(0.5 * TrueAnomaly * Pi/180.0)
                                  * sqrt((1.0 - e) /(1.0 + e))   );
   Isun := a*PiSun*(cos(EccentricAnomaly)-e)+
             a*sqrt(1.0-e*e)*QiSun*sin(EccentricAnomaly);
   Jsun := a*PjSun*(cos(EccentricAnomaly)-e)+
             a*sqrt(1.0-e*e)*QjSun*sin(EccentricAnomaly);
   Ksun := a*PkSun*(cos(EccentricAnomaly)-e)+
             a*sqrt(1.0-e*e)*QkSun*sin(EccentricAnomaly);
   Rsun := sqrt(Isun*Isun + Jsun*Jsun + Ksun*Ksun); (* in AU *)

 end;  (* CalculateObjPosition *)


procedure CalculateObjVelocity;

(* Velocities are in AU's *)

var
  parameter: real;
  sqrtHp,sinT,cosT: real;  (* temporaries *)

begin
```

```
  sinT   := sin(TrueAnomaly * Pi/180.0);
  cosT   := cos(TrueAnomaly * Pi/180.0);

  parameter := a * (1.0 - e * e);

  sqrtHp := sqrt(HelioGravConst / parameter) ;


      VelocI := sqrtHp * (  (-sinT * PiSun)   +   (e + cosT) * QiSun );
      VelocJ := sqrtHp * (  (-sinT * PjSun)   +   (e + cosT) * QjSun );
      VelocK := sqrtHp * (  (-sinT * PkSun)   +   (e + cosT) * QkSun );
end; (* CalculateObjVelocity *)



procedure Departure360;
  var
    counter: integer;

    begin
      writeln;
      writeln;
      CalcAuxQuant;
      TrueAnomaly := 0.0;
         while TrueAnomaly <= 360.0 do
            begin
              counter := round(TrueAnomaly/10.0);
              CalculateObjPosition;
              CalculateObjVelocity;
              DeparturePosition [counter,1] := Isun * AU ;
              DeparturePosition [counter,2] := Jsun * AU ;
              DeparturePosition [counter,3] := Ksun * AU ;


              DepartureVelocity [counter,1] := VelocI * AU;
              DepartureVelocity [counter,2] := VelocJ * AU;
              DepartureVelocity [counter,3] := VelocK * AU;

              TrueAnomaly := TrueAnomaly + 10.0;
            end; (* while *)
      end;  (* Departure360 *)

procedure Arrival360;
  var
    counter: integer;

    begin
      writeln;
      writeln;
      CalcAuxQuant;
      TrueAnomaly := 0.0;
         while TrueAnomaly <= 360.0 do
```

```
            begin
              counter := round(TrueAnomaly/10.0);
              CalculateObjPosition;
              CalculateObjVelocity;
              ArrivalPosition [counter,1] := Isun * AU ;
              ArrivalPosition [counter,2] := Jsun * AU ;
              ArrivalPosition [counter,3] := Ksun * AU ;


              ArrivalVelocity [counter,1] := VelocI * AU;
              ArrivalVelocity [counter,2] := VelocJ * AU;
              ArrivalVelocity [counter,3] := VelocK * AU;

              TrueAnomaly := TrueAnomaly + 10.0;
            end; (* while *)
        end;  (* Arrival360 *)


procedure R1crossR2;

begin

  R1xR2[1] :=     DepartPos[2] * ArrivalPos[3]
                - DepartPos[3] * ArrivalPos[2]  ;

  R1xR2[2] :=     DepartPos[3] * ArrivalPos[1]
                - DepartPos[1] * ArrivalPos[3]  ;

  R1xR2[3] :=     DepartPos[1] * ArrivalPos[2]
                - DepartPos[2] * ArrivalPos[1]  ;


  MagR1xR2 := sqrt( R1xR2[1] * R1xR2[1]
                  + R1xR2[2] * R1xR2[2]
                  + R1xR2[3] * R1xR2[3] ) ;

end; (* R1crossR2 *)



procedure AngleR1R2;
var
  hearth: array[1..3] of real:
  temp: real;

begin
   AngleBetweenVectors := arccos(  ( DepartPos[1] * ArrivalPos[1]
                                   + DepartPos[2] * ArrivalPos[2]
                                   + DepartPos[3] * ArrivalPos[3] )
                                             / ( r1 * r2 ) );
  R1crossR2;
```

```
            (*  calculate q   *)

    hearth[1] :=     DepartPos[2] * DepartVel[3]
                   - DepartPos[3] * DepartVel[2]  ;

    hearth[2] :=     DepartPos[3] * DepartVel[1]
                   - DepartPos[1] * DepartVel[3]  ;

    hearth[3] :=     DepartPos[1] * DepartVel[2]
                   - DepartPos[2] * DepartVel[1]  ;


      temp :=   R1xR2[1] * hearth[1]
              + R1xR2[2] * hearth[2]
              + R1xR2[3] * hearth[3]  ;

      if temp >= 0.0 then q  := 1 else q  := -1 ;

      If q  = -1 then AngleBetweenVectors := 2.0 * Pi
                                              - AngleBetweenVectors;
  end;  (* AngleR1R2 *)


  procedure GetNextSetTrueAnomalies;
     var
        counter : integer;
     begin
       for counter := 1 to 3 do
           begin
             DepartPos[counter] := DeparturePosition[TrueEarth,counter];
          ArrivalPos[counter]   := ArrivalPosition[TrueTarget,counter];
          DepartVel[counter]    := DepartureVelocity[TrueEarth,counter];
         ArrivalVel[counter]    := ArrivalVelocity[TrueTarget,counter];
           end;

            r1 :=      sqrt(DepartPos[1] * DepartPos[1]
                           +DepartPos[2] * DepartPos[2]
                           +DepartPos[3] * DepartPos[3]);

            r2 :=      sqrt(ArrivalPos[1] * ArrivalPos[1]
                           +ArrivalPos[2] * ArrivalPos[2]
                           +ArrivalPos[3] * ArrivalPos[3]);

               U1[1] := DeparturePosition [TrueEarth,1] / r1;
               U1[2] := DeparturePosition [TrueEarth,2] / r1;
               U1[3] := DeparturePosition [TrueEarth,3] / r1;

               U2[1] := ArrivalPosition [TrueTarget,1] / r2;
               U2[2] := ArrivalPosition [TrueTarget,2] / r2;
               U2[3] := ArrivalPosition [TrueTarget,3] / r2;

     end; (* proc GetNextSetTrueAnomalies *)
```

```
procedure Shade(X,Y: integer; c : real);
var
s,a,b,d,e: integer;

begin

if c <= 4.5 then
  begin
  for s := 0 to 2 do
  begin
  for a := 0 to 3 do
    begin
      for b := 0 to 3 do
        begin
          plot(x+s*4+a,y+b,1);
        end;
    end;
  end;
  end;


if (c <= 5.0) and (c > 4.5) then
   begin
     for s := 0 to 2 do
        begin
          plot(x+s*4+1,y,1);
          plot(x+s*4+2,y,1);
          plot(x+s*4+3,y,1);
          plot(x+s*4,y+1,1);
          plot(x+s*4+2,y+1,1);
          plot(x+s*4+3,y+1,1);
          plot(x+s*4,y+2,1);
          plot(x+s*4+1,y+2,1);
          plot(x+s*4+3,y+2,1);
          plot(x+s*4,y+3,1);
          plot(x+s*4+1,y+3,1);
          plot(x+s*4+2,y+3,1);
         end;
    end;


if (c <= 6.0) and (c > 5.0) then
    begin
      for s := 0 to 2 do
         begin
          plot(x+s*4,y,1);
          plot(x+s*4+3,y,1);
          plot(x+s*4+1,y+1,1);
          plot(x+s*4+2,y+1,1);
          plot(x+s*4+1,y+2,1);
          plot(x+s*4+2,y+2,1);
          plot(x+s*4,y+3,1);
```

```
          plot(x+s*4+3,y+3,1);
         end;
    end;

if (c <= 8.0) and (c > 6.0) then
    begin
      for s := 0 to 2 do
        begin
         plot(x+s*4,y,1);
         plot(x+s*4+3,y,1);
         plot(x+s*4+1,y+1,1);
         plot(x+s*4+2,y+2,1);
         plot(x+s*4,y+3,1);
         plot(x+s*4+3,y+3,1);
        end;
    end;

if (c <= 10.0) and (c > 8.0) then
    begin
      for s := 0 to 2 do
        begin
         plot(x+s*4,y,1);
         plot(x+s*4+2,y+1,1);
         plot(x+s*4+1,y+2,1);
         plot(x+s*4+2,y+4,1);
        end;
    end;

if (c <= 14.0) and (c > 10.0) then
    begin
      for s := 0 to 2 do
        begin
         plot(x+s*4+1,y+1,1);
         plot(x+s*4+2,y+2,1);
       end;
    end;

if (c  < 20.0) and (c > 14.0) then
    begin
      for s := 0 to 2 do
        begin
         plot(x+s*4+2,y+2,1);
        end;
    end;
end; (* Shade *)

procedure CalcI;
var
                  V : array[1..3] of real;
                 v1 : array[1..3] of real;
                 v2 : array[1..3] of real;
                 p1 : array[1..3] of real;
```

```
                     p2 : array[1..3] of real;
                     I1 : array[1..3] of real;
                     I2 : array[1..3] of real;
                    dI1 : array[1..3] of real;
                    dI2 : array[1..3] of real;

                dV1,dV2 : real;
                      z : real;
               p,pn,pn1 : real;
             pmin, pmax : real;
                sqrtupt : real;
                     SL : real;
     TempQq1, TempQq2 : real;
                 r1dotr2 : real;
                I1dotdI1 : real;
                I2dotdI2 : real;
 TempNumer, TempDenom : real;
                  Theta : real;
               Theta360 : real;
                      t : real;
               J, TempJ : real;
                 sqrtup : real;
                 I1dotI1 : real;
                 I2dotI2 : real;
                   Diff : real;
                  Tempp : real;

const
   u = 132718E6;   (* km3/s2 *)
  q1 = 6656.0;
  q2 = 15.0;
  QQ1 = 6656.0;
  QQ2 = 5E1;
   uu1 = 3.986005E5;  (* km3/s2 *)
   uu2 = 3.37E-7;  (* km3/s2 Direct proportion to volume 1982DB
                                                 / volume Earth  *)


Procedure FindDiff;

begin
         sqrtup := sqrt(u * p);
              z := sqrt(u / p) * tan(Theta/2.0);
 for counter := 1 to 3 do
   begin
   v[counter] := sqrtup * (p2[counter] - p1[counter] ) / MagR1xR2;

   I1[counter] := - v1[counter] + SL * ( v[counter] + Z * U1[counter]);
   I2[counter] :=   v2[counter] - SL * ( v[counter] - Z * U2[counter]);
  dI1[counter] :=  SL * 0.5 / p * (v[counter] - Z * U1[counter]);
  dI2[counter] := -SL * 0.5 / p * (v[counter] + z * U2[counter]);
   end;
```

```
   TempQq1 := sqrt(2.0 * uu1 * QQ1/(q1*(QQ1+q1)) ) ;
   TempQq2 := sqrt(2.0 * uu2 * QQ2/(q2*(QQ2+q2)) ) ;

   I1dotI1 := I1[1] * I1[1] + I1[2] * I1[2] + I1[3] * I1[3];
   I2dotI2 := I2[1] * I2[1] + I2[2] * I2[2] + I2[3] * I2[3];

   dV1 := sqrt( I1dotI1 + (2.0 * uu1/q1) )  - TempQq1 ;
   dV2 := sqrt( I2dotI2 + (2.0 * uu2/q2) )  - TempQq2 ;

   I1dotdI1 := I1[1] * dI1[1] + I1[2] * dI1[2] + I1[3] * dI1[3];
   I2dotdI2 := I2[1] * dI2[1] + I2[2] * dI2[2] + I2[3] * dI2[3];

Diff :=  I1dotdI1  / ( dV1 + TempQq1 )   +    I2dotdI2
                                            / ( dV2 + TempQq2 ) ;

end; (* FindDiff *)


procedure Newton;
var
  dVee, dVeen, k : real;
  Deltap : real;
 dVeeOld : real;
       c : integer;
const
  accuracy = 0.0001;

begin
      Deltap := 10.0;
        Dvee := 100.0;
        dVeeOld:= 1000.0;

        p := (pMax + pMin)/2.0;
        k := 0.5 * p;
        c := 0;
     while (abs(dVee-dVeeOld) >  accuracy) do

       begin
        c := c + 1;
        dVeeOld := dVee;
        FindDiff;
        dVee := dV1+dV2;
        p := p+Deltap;
        FindDiff;
        dVeen := dV1+dV2;
        if dVeen > dVee then
          begin
            k := -k;
            Deltap := -Deltap;
          end;
        k := k/2.0;
        p := p + k ;
```

```
      if c > 50 then dVee := dVeeOld;
     end;

  J := dV1+dV2;

end; (* Newton *)



procedure GaussGodalTimeOfFlight(r1,r2,Theta,p: real; var t: real);
const
  u = 123718E15;  (* m3/s2 *)
var
  sinHalfTheta : real;
  cosHalfTheta : real;
  cosz, sinz   : real;
  z            : real;
  A, B         : real;
        Bmcosz : real;

begin
        r1 := r1 * 1000.0;
        r2 := r2 * 1000.0;
         p := p  * 1000.0;

        sinHalfTheta := sin(Theta/2.0) ;
        cosHalfTheta := cos(Theta/2.0) ;

  cosz :=  (-2.0 * r1 * r2 * sinHalfTheta * sinHalfTheta/p + r1 + r2)
            /2.0/sqrt(r1)/sqrt(r2)/cosHalfTheta ;

Bmcosz :=    sqrt(r1)*sqrt(r2) * sinHalfTheta * sinHalfTheta / p    /
                 cosHalfTheta   ;

   z := arccos(cosz);
   sinz := sin(z);
  A := 2.0 / sqrt(u) * power(r1,0.75)*power(r2,0.75) *
                      power(abs(cosHalfTheta),1.5) ;
  B := (r1 + r2) / 2.0 / sqrt(r1) / sqrt(r2) / cosHalfTheta ;

  t := A * sqrt(abs(Bmcosz)) * (1.0 +   (Bmcosz) * (2.0*z - sin(2.0*z))/
                                     2.0 / sinz / sinz / sinz );
  t := abs(t / 3600.0 / 24.0) ;

end; (* GaussGodelTimeOfFlight *)


begin
     for counter := 1 to 3 do
        begin
           p1[counter] := DepartPos[counter];
           p2[counter] := ArrivalPos[counter];
```

```
          v1[counter] := DepartVel[counter];
          v2[counter] := ArrivalVel[counter];
       end;

       Theta := AngleBetweenVectors ;
       Theta360 := Theta;
       if theta > Pi then theta := 2.0 * Pi - theta;

    r1dotr2 := p1[1] * p2[1]  +  p1[2] * p2[2] + p1[3] * p2[3] ;

    TempNumer := r1 * r2 - r1dotr2;
    TempDenom := sqrt( 2.0 * (r1 * r2 + r1dotr2) ) ;

    pmin := tempNumer / (r1 + r2 + TempDenom);
    pmax := tempNumer / (r1 + r2 - TempDenom);

    if Theta360 > Pi then SL := -1.0 else SL := 1.0 ;
    Newton;
    GaussGodalTimeOfFlight(r1,r2,Theta360,p,t);

    DeltaVeeStore[TrueEarth,TrueTarget] := J;
    TimeOfFltStore[TrueEarth,TrueTarget] := t;

(* writeln(10 * TrueEarth:7,10 * TrueTarget:7,J:10:4,'  ',t:6:1); *)

    shade(85 + TrueTarget*12,45 + TrueEarth*4,J);


end; (* CalcI *)


procedure DeltaV360;
  var
     count  : integer;
     counter: integer;
  begin
     for count := 0 to 36 do
       begin
         TrueEarth := count ;
         for counter := 0 to 36 do
          begin
            TrueTarget := counter;
            GetNextSetTrueAnomalies;
            AngleR1R2;

            CalcI;

          end; (* counter *)
       end; (* count *)

   end; (* proc DeltaV360 *)
```

```
procedure InitOutFile;
begin
  writeln; write('What is the name (prefix) of the output file ?  ');
  readln(OutFile);
  assign(DataOut, OutFile+'.OUT');
  rewrite(DataOut);
end; (* InitOutFile *)

procedure WriteDVFile;
var
  i,page,column,row : integer;
begin
   for page := 0 to 2 do
     begin
       write(DataOut,'Arr®Dep');
       for i := 0 to 11 do
        begin
          write(DataOut,(i+12*page)*10:5);
        end; (* for i *)
        writeln(DataOut);

       for row := 36 downto 0 do
         begin
           write(DataOut,row*10:3,'    ');
           for column := 0 + 12*page to 11 + 12*page do
             begin
               write(DataOut,DeltaVeeStore[column,row]:5:1);
             end; (* for column *)
           writeln(DataOut);
         end; (* for row *)
         writeln(DataOut); writeln(DataOut); writeln(DataOut);
      end; (* for page *)

end; (* WriteDVOut *)


procedure WriteTOFFile;
var
  i,page,column,row : integer;
begin
   for page := 0 to 2 do
     begin
       write(DataOut,'Arr®Dep');
       for i := 0 to 11 do
        begin
          write(DataOut,(i+12*page)*10:5);
        end; (* for i *)
        writeln(DataOut);

       for row := 36 downto 0 do
         begin
```

```
            write(DataOut,row*10:3,'    ');
            for column := 0 + 12*page to 11 + 12*page do
              begin
                write(DataOut,TimeOfFltStore[column,row]:5:0);
              end; (* for column *)
            writeln(DataOut);
          end; (* for row *)
          writeln(DataOut); writeln(DataOut); writeln(DataOut);
      end; (* for page *)

end; (* WriteTOFOut *)

procedure InitPrimeRibs;
var
  i : integer;
begin
  draw( 74, 193, 539, 193, 1);
  draw( 74,  44, 539,  44, 1);
  draw( 84, 197,  84,  40, 1);
  draw(529, 197, 529,  40, 1);

  for i := 1 to 8 do
    begin
      draw( 84+48*i, 193,  84+48*i,  197, 1);
      draw( 84+48*i,  44,  84+48*i,   40, 1);
      draw(  84,  44+16*i,   74, 44+16*i, 1);
      draw( 529,  44+16*i,  539, 44+16*i, 1);
    end; (* for i *)
end; (* InitPrimeRibs *)


begin (* main *)
  InitialDisplay;
  GetDataFile;
  SelectObject;
  ConvertToRadians;
  PreComputeTranscendentals;
  Departure360;
  InitialDisplay;
  GetDataFile;
  SelectObject;
  ConvertToRadians;
  PreComputeTranscendentals;
  Arrival360;

  InitOutFile;
  ClrScr;
  HiRes;
  HiResColor(7);
  InitPrimeRibs;
  DeltaV360;
```

```
  readln;
  WriteDVFile;
  WriteTOFFile;
  close(DataOut);
  writeln('END END');

  readln; (* return to DOS *)
end. (* main *)
```

Appendix B.

PASCAL Program to Implement
the Ross-Hulkower Algorithms

```
program RossHulkowerDeltaVee;

(* Ross-Kulkower algorithm to find optimal delta vee. Calculates     *)
(* objects position and velocity in heliocentric/ecliptic IJK. ref   *)
(* frame. Then runs DeltaVee program on selected combinations of     *)
(* Departure/Arrival  True Anomalies.                                *)

const
   MaxRecSize = 25;
   HelioGravConst = 3.9640157489E-14;   (* AU3/s2  derived from
                                        Astronomical Almanac 85 p. K6 *)
   AU = 1.49597870E8;                   (* km      derived from
                                        Astronomical Almanac 85 p. K6 *)
type
    Name = string [10];
    ElementSet = record
          AsteroidName: Name;   (* object name *)
                    Tp: real;   (* date of perihelion passage/epoch *)
                    i : real;   (* inclination to ecliptic *)
                    w : real;   (* arguement of perihelion *)
                    N : real;   (* longitude of ascending node *)
                    a : real;   (* semi-major axis, in AU  *)
                    e : real;   (* eccentricity *)
                    Mo: real;   (* mean anomoly at To,
                                           Mo = 0 at perihelion *)
                end;
var
     AsteroidFile: text;
     FileName: string[14];
     ElementSetRec: array [1..MaxRecSize] of ElementSet;
     Ifor: integer;
     MaxNoOfAsteroids: integer;
     Selected: integer;  (* record selected to work with *)

          AsteroidName: name;
                    Tp: real;   (* date of perihelion passage/epoch *)
                    i : real;   (* inclination to ecliptic *)
                    w : real;   (* arguement of perihelion *)
                    N : real;   (* longitude of ascending node *)
                    a : real;   (* semi-major axis, in AU  *)
                    e : real;   (* eccentricity *)
                    Mo: real;   (* mean anomoly at To,
                                           Mo = 0 at perihelion *)
     TrueAnomaly, EccentricAnomaly: real;
     Isun, Jsun, Ksun: real;      (* heliocentric/ecliptic coord *)
```

```
    VelocI, VelocJ, VelocK: real;(* heliocentric/ecliptic velocity *)
    Rsun: real;                  (* distance from sun *)
    A1,A2,B1,B2,Y1,Y2: real;     (* auxiliaries *)
    PiSun, PjSun, PkSun, QiSun, QjSun, QkSun: real; (* auxiliaries *)
    sini, sinw, sinN: real;  (* to precompute sin(i) etc.  *)
    cosi, cosw, cosN: real;

    DeparturePosition:     array[0..35,1..3] of real;
    DepartureVelocity:     array[0..35,1..3] of real;
    ArrivalPosition:       array[0..35,1..3] of real;
    ArrivalVelocity:       array[0..35,1..3] of real;

    icPir1,icMir1,icPir2,icMir2,h:                 array[1..3] of real;
    UniticPir1,UniticMir1,UniticPir2,UniticMir2:  array[1..3] of real;

    r1,r2:real;
    CC : real;
    q, SG : real;
    P1,P2,M1,M2: real;

    DepartVelocOld:     array[1..3] of real;  (* old coord sys *)
    DepartVelocNew:     array[1..3] of real;  (* old coord sys *)
    ArriveVelocOld:     array[1..3] of real;  (* old coord sys *)
    ArriveVelocNew:     array[1..3] of real;  (* old coord sys *)

    DeparturePositOld:  array[1..3] of real;  (* old coord sys *)
    ArrivalPositOld:    array[1..3] of real;  (* old coord sys *)
    DeparturePositNew:  array[1..3] of real;  (* new coord sys *)
    ArrivalPositNew:    array[1..3] of real;  (* new coord sys *)

    R1xR2: array[1..3] of real;

    AngleBetweenVectors: real;

    counter: integer;
    TrueStar, TrueP : integer;


function tan(x: real) : real;
   begin
     tan := sin(x)/cos(x);
   end; (* tan *)

        (* *** The following 8 procedures are identical *** *)
        (* *** to procedures of the same names in the ***** *)
        (* *** program HulkowerLauBenderDeltaVee ********** *)

procedure GetDataFile;  (* Reads data into global variables *)

procedure SelectObject; (* Global variables *)

procedure InitialDisplay;
```

```
procedure ConvertToRadians;

procedure PreComputeTranscendentals;

procedure CalcAuxQuant ;

procedure CalculateObjPosition;

procedure CalculateObjVelocity;


procedure Departure360;

   (*  This procedure is modified to operate on a single true     *)
   (*  anomaly rather than the full range of 0-360 because of     *)
   (*  the instability of the algorithms. Problems of             *)
   (*  convergence may occur at some values of departure true     *)
   (*  anomalies. The procedure will operate through the 0-360    *)
   (*  if it is reconfigured like the Departure360 algorithm      *)
   (*  in the program HulkowerLauBenderDeltaVee.                  *)

  var
    counter: integer;

    begin
      writeln;
      writeln;
      CalcAuxQuant;
      TrueAnomaly := 110.0;
         while TrueAnomaly <= 110.0 do
            begin
              counter := round(TrueAnomaly/10.0);
              CalculateObjPosition;
              CalculateObjVelocity;
              DeparturePosition [counter,1] := Isun;
              DeparturePosition [counter,2] := Jsun;
              DeparturePosition [counter,3] := Ksun;

              DepartureVelocity [counter,1] := VelocI * AU;
              DepartureVelocity [counter,2] := VelocJ * AU;
              DepartureVelocity [counter,3] := VelocK * AU;

              TrueAnomaly := TrueAnomaly + 10.0;
            end; (* while *)
     end;  (* Departure360 *)

procedure Arrival360;
  var
    counter: integer;
    begin
      writeln;
```

```
        writeln;
        CalcAuxQuant;
        TrueAnomaly := 0.0;
           while TrueAnomaly <= 360.0 do

              begin
                counter := round(TrueAnomaly/10.0);
                CalculateObjPosition;
                CalculateObjVelocity;
                ArrivalPosition [counter,1] := Isun;
                ArrivalPosition [counter,2] := Jsun;
                ArrivalPosition [counter,3] := Ksun;

                ArrivalVelocity [counter,1] := VelocI * AU;
                ArrivalVelocity [counter,2] := VelocJ * AU;
                ArrivalVelocity [counter,3] := VelocK * AU;

                TrueAnomaly := TrueAnomaly + 10.0;
              end; (* while *)
       end;  (* Arrival360 *)


procedure GetAlphaBetaGamma;
var
  hstar : array[1..3] of real;
  temp  : real;

begin
  DepartVelocNew[1] :=   DepartVelocOld[1] * UniticPir1[1]
                       + DepartVelocOld[2] * UniticPir1[2]
                       + DepartVelocOld[3] * UniticPir1[3]  ;

  DepartVelocNew[2] :=   DepartVelocOld[1] * UniticMir1[1]
                       + DepartVelocOld[2] * UniticMir1[2]
                       + DepartVelocOld[3] * UniticMir1[3]  ;

  DepartVelocNew[3] :=   DepartVelocOld[1] * h[1]
                       + DepartVelocOld[2] * h[2]
                       + DepartVelocOld[3] * h[3]  ;


  ArriveVelocNew[1] :=   ArriveVelocOld[1] * UniticMir2[1]
                       + ArriveVelocOld[2] * UniticMir2[2]
                       + ArriveVelocOld[3] * UniticMir2[3]  ;

  ArriveVelocNew[2] :=   ArriveVelocOld[1] * UniticPir2[1]
                       + ArriveVelocOld[2] * UniticPir2[2]
                       + ArriveVelocOld[3] * UniticPir2[3]  ;

  ArriveVelocNew[3] :=   ArriveVelocOld[1] * h[1]
                       + ArriveVelocOld[2] * h[2]
                       + ArriveVelocOld[3] * h[3]  ;
```

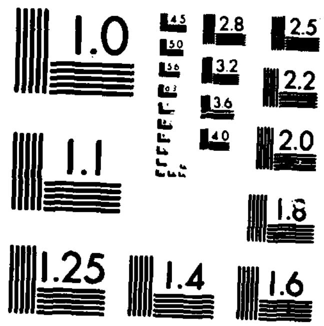
```
          DepartVelocNew[1] := DepartVelocNew[1]  * P1;
          DepartVelocNew[2] := DepartVelocNew[2]  * M1;
          ArriveVelocNew[1] := ArriveVelocNew[1]  * M2;
          ArriveVelocNew[2] := ArriveVelocNew[2]  * P2;

           (*  calculate q   *)

  hstar[1] :=     DeparturePositOld[2] * DepartVelocOld[3]
                - DeparturePositOld[3] * DepartVelocOld[2]  ;

  hstar[2] :=     DeparturePositOld[3] * DepartVelocOld[1]
                - DeparturePositOld[1] * DepartVelocOld[3]  ;

  hstar[3] :=     DeparturePositOld[1] * DepartVelocOld[2]
                - DeparturePositOld[2] * DepartVelocOld[1]  ;


      temp :=   R1xR2[1] * hstar[1]
              + R1xR2[2] * hstar[2]
              + R1xR2[3] * hstar[3]  ;
      if temp >= 0.0 then q  := 1.0 else q  := -1.0 ;

end; (* GetAlphaBetaGamma *)


procedure UnitVectorh;
var
  d: real;

begin
      d := r1 * r2 * sin(AngleBetweenVectors) ;

  R1xR2[1] :=     DeparturePositOld[2] * ArrivalPositOld[3]
                - DeparturePositOld[3] * ArrivalPositOld[2]  ;

  R1xR2[2] :=     DeparturePositOld[3] * ArrivalPositOld[1]
                - DeparturePositOld[1] * ArrivalPositOld[3]  ;

  R1xR2[3] :=     DeparturePositOld[1] * ArrivalPositOld[2]
                - DeparturePositOld[2] * ArrivalPositOld[1]  ;

      h[1] :=  R1xR2[1] / d ;
      h[2] :=  R1xR2[2] / d ;
      h[3] :=  R1xR2[3] / d ;

end; (* UnitVectorh *)
```

```
function arccos(a:real): real;
   var
     temp: real;

   begin
      if a = 0.0 then a := 0.0000000001;      (* avoid divide by zero *)
      temp := arctan(sqrt(1.0 - a * a) / a);
      if temp < 0.0 then arccos := temp + Pi else arccos := temp;
   end;



procedure AngleR1R2;
begin
   AngleBetweenVectors := arccos(
                         ( DeparturePositOld[1] * ArrivalPositOld[1]
                         + DeparturePositOld[2] * ArrivalPositOld[2]
                         + DeparturePositOld[3] * ArrivalPositOld[3] )
                                                   / ( r1 * r2 )   );
end;  (* AngleR1R2 *)


procedure UnitVectorcr1r2;
var
   ir1, ir2, ic, c: array[1..3] of real;
   cx : real;
   temp : real;
begin
   ir1[1] := DeparturePositOld[1] / r1;
   ir1[2] := DeparturePositOld[2] / r1;
   ir1[3] := DeparturePositOld[3] / r1;

   ir2[1] := ArrivalPositOld[1] / r2;
   ir2[2] := ArrivalPositOld[2] / r2;
   ir2[3] := ArrivalPositOld[3] / r2;

   c[1] := ArrivalPositOld[1] - DeparturePositOld[1];
   c[2] := ArrivalPositOld[2] - DeparturePositOld[2];
   c[3] := ArrivalPositOld[3] - DeparturePositOld[3];

   cx := sqrt  (  c[1] * c[1]   +    c[2] * c[2]   +    c[3] * c[3]  );
   CC := cx;

   ic[1] := c[1] / cx;
   ic[2] := c[2] / cx;
   ic[3] := c[3] / cx;

   icPir1[1] := ic[1] + ir1[1];
   icPir1[2] := ic[2] + ir1[2];
   icPir1[3] := ic[3] + ir1[3];

temp := sqrt(icPir1[1] * icPir1[1] + icPir1[2] * icPir1[2]
```

```
                                     + icPir1[3] * icPir1[3]);
P1 := temp;

       UniticPir1[1] := icPir1[1] / temp ;
       UniticPir1[2] := icPir1[2] / temp ;
       UniticPir1[3] := icPir1[3] / temp ;

   icMir1[1] := ic[1] - ir1[1];
   icMir1[2] := ic[2] - ir1[2];
   icMir1[3] := ic[3] - ir1[3];

temp := sqrt(icMir1[1]*icMir1[1] + icMir1[2]*icMir1[2]
                                 + icMir1[3]*icMir1[3]);

       UniticMir1[1] := icMir1[1] / temp ;
       UniticMir1[2] := icMir1[2] / temp ;
       UniticMir1[3] := icMir1[3] / temp ;

   M1 := temp;

   icPir2[1] := ic[1] + ir2[1];
   icPir2[2] := ic[2] + ir2[2];
   icPir2[3] := ic[3] + ir2[3];

temp := sqrt(icPir2[1]*icPir2[1] + icPir2[2]*icPir2[2]
                                 + icPir2[3]*icPir2[3]);

       UniticPir2[1] := icPir2[1] / temp ;
       UniticPir2[2] := icPir2[2] / temp ;
       UniticPir2[3] := icPir2[3] / temp ;

   P2 := temp;

   icMir2[1] := ic[1] - ir2[1];
   icMir2[2] := ic[2] - ir2[2];
   icMir2[3] := ic[3] - ir2[3];

temp := sqrt(icMir2[1]*icMir2[1] + icMir2[2]*icMir2[2]
                                 + icMir2[3]*icMir2[3]);

       UniticMir2[1] := icMir2[1] / temp ;
       UniticMir2[2] := icMir2[2] / temp ;
       UniticMir2[3] := icMir2[3] / temp ;

   M2 := temp;

end;  (* UnitVectorcr1r2 *)
```

AD-A163 976 AN ANALYSIS OF THE ACCESIBILITY OF EARTH-APPROACHING ASTEROIDS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING P W SOMERS DEC 85 2/2

UNCLASSIFIED AFIT/GSO/AA/85D-1 F/G 22/3 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
procedure GetNextSetTrueAnomalies;
   var
      counter : integer;
   begin
     for counter := 1 to 3 do
         begin
     DeparturePositOld[counter] := DeparturePosition[TrueStar,counter];
     ArrivalPositOld[counter]   := ArrivalPosition[TrueP,counter];
     DepartVelocOld[counter]    := DepartureVelocity[TrueStar,counter];
     ArriveVelocOld[counter]    := ArrivalVelocity[TrueP,counter];
         end;

          r1 := sqrt(departurepositold[1] * departurepositold[1]
                    +departurepositold[2] * departurepositold[2]
                    +departurepositold[3] * departurepositold[3]);

          r2 := sqrt(arrivalpositold[1] * arrivalpositold[1]
                    +arrivalpositold[2] * arrivalpositold[2]
                    +arrivalpositold[3] * arrivalpositold[3]);

   end; (* proc GetNextSetTrueAnomalies *)


procedure FindDeltaV( AlphaStar, BetaStar, GammaStar,
            AlphaP, BetaP, GammaP, C, q , SG, r1, r2, Theta: real);

var
  DeltaV1, DeltaV2: real;
  sqrDeltaV1, sqrDeltaV2: real;
  oldk1,oldk2: real;
  GravConstStar, GravConstP: real;
  GravConstEarth: real;
  K, K1, K2: real;
  Epsilon: real;
  TempEpsilon: real;
  s: real;
  GravConstSun: real;


function tan(a: real): real;
begin
   tan := sin(a)/cos(a);
end;



procedure FindEpsilon;
var
  accuracy: real;
  Epsilon1, Epsilon2: real;
```

```
procedure PreEpsilon;
begin

  Epsilon1 := q /(2.0 * K *  (k1 - k2))  *

      (     k1 * AlphaStar * (1.0 +   (r2 * cos(theta))/C  -  r1/C )
          + k2 * AlphaP    * (1.0 +   (r1 * cos(theta))/C  -  r2/C ) );

  Epsilon2 := SG / (2.0 * K * (k1 - k2))  *

      (     k1 * BetaStar  * (1.0 -   (r2 * cos(theta))/C  +  r1/C )
          + k2 * BetaP     * (1.0 -   (r1 * cos(theta))/C  +  r2/C ) );

end; (* PreEpsilon *)

begin  (* FindEpsilon *)

  accuracy := 0.00001;
  PreEpsilon;
  repeat
      TempEpsilon := ABS(Epsilon) ;
      Epsilon := arctan(ABS(Epsilon1 * sin(TempEpsilon) + Epsilon2));

   write('*');
   until (abs(TempEpsilon - Epsilon) < accuracy);

end; (* FindEpsilon *)



begin (* main procedure FindDeltaV *)

  TempEpsilon := 1.0;   (* artificial starting value *)
  Epsilon := -0.01;       (* artificial starting value *)

   GravConstSun := 132718E6;
   GravConstEarth := 398603.0;

   GravConstStar := GravConstEarth;

   k1 := 0.8;
   k2 := 0.7;

   s := 0.5 * (r1 + r2 + C);
   k := sqrt(   (GravConstSun * C) / (2.0 * s * (s - C) )  );

     FindEpsilon;
       oldk1 := -200.0;

repeat
```

```
     sqrDeltaV1 :=
        (K * q  / cos(Epsilon) - AlphaStar )
         * (K * q  / cos(Epsilon) - AlphaStar )
      + (K * SG * tan(Epsilon) - BetaStar )
         * (K * SG * tan(Epsilon) - BetaStar )
      + (GammaStar * GammaStar)  ;

     sqrDeltaV2 :=
        (K * q  / cos(Epsilon) - AlphaP )
        * (K * q  / cos(Epsilon) - AlphaP )
      + (K * SG * tan(Epsilon) - BetaP )
      * (K * SG * tan(Epsilon) - BetaP )
      + (GammaP * GammaP)  ;

      oldk1 := k1 ;

      k1 := 1.0 / sqrt( sqrDeltaV1 + (2.0 * GravConstStar / 6700.0) );
      k2 := -1.0 / sqrt( sqrDeltaV2 );

      FindEpsilon;

   until  (abs(oldk1 - k1) < 0.00001);

     sqrDeltaV1 :=
        (K * q  / cos(Epsilon) - AlphaStar )
         * (K * q  / cos(Epsilon) - AlphaStar )
         / P1 / P1
      + (K * SG * tan(Epsilon) - BetaStar )
         * (K * SG * tan(Epsilon) - BetaStar )
         / M1 / M1
      + (GammaStar * GammaStar)  ;


     sqrDeltaV2 :=
        (K * q  / cos(Epsilon) - AlphaP )
        * (K * q  / cos(Epsilon) - AlphaP )
        / M2 / M2
      + (K * SG * tan(Epsilon) - BetaP )
      * (K * SG * tan(Epsilon) - BetaP )
        / P2 / P2
      + (GammaP * GammaP)  ;

writeln(
TrueStar*10:4,TrueP*10:4,sqrt(sqrDeltaV1):12:2,sqrt(sqrDeltaV2):12:2);

writeln('END');

end; (* proc FindDeltaV1 *)
```

```
procedure DeltaV360;
  var
     counter: integer;
  begin
     TrueStar := 11 ;
     write('SG = ');
     readln(sg);
     for counter := 0 to 360 do
       begin
         TrueP := counter;
         GetNextSetTrueAnomalies;
         UnitVectorcrlr2;
         AngleR1R2;
         UnitVectorh;
         GetAlphaBetaGamma;

         FindDeltaV(
           DepartVelocNew[1],DepartVelocNew[2],DepartVelocNew[3],
           ArriveVelocNew[1],ArriveVelocNew[2],ArriveVelocNew[3],
           CC * AU,q ,SG,r1 * AU,r2 * AU,AngleBetweenVectors);

       end; (* for counter *)

end; (* proc DeltaV360 *)


begin (* main *)
  InitialDisplay;
  GetDataFile;
  SelectObject;
  ConvertToRadians;
  PreComputeTranscendentals;
  Departure360;
  InitialDisplay;
  GetDataFile;
  SelectObject;
  ConvertToRadians;
  PreComputeTranscendentals;
  Arrival360;
  ClrScr;
  DeltaV360;

  writeln;
  writeln;
  readln; (* return to DOS *)
end. (* main *)
```

Appendix C.

PASCAL Program to Generate the Validation Ephemerides

```
program HelioCentricOrbit2050;

(* This program calculates ephemerides for solar system objects. It   *)
(* uses the following data files: PLANET.ELE, COMET.ELE, AST-0001.ELE *)
(* AAA82-84.ELE and any user-defined data file with the same format.  *)
(* Either a single date or a period ephemeris can be output.          *)

const
   MaxRecSize = 25;
   ObliqEclip = 23.439;
   nAdjust1985 = -5479.5;
   JD00Jan1985 = 2446065.5;

type
    Name = string [10];
    ElementSet = record
          AsteroidName: Name;    (* object name *)
                    Tp: real;    (* date of perihelion passage/epoch *)
                    i : real;    (* inclination to ecliptic *)
                    w : real;    (* arguement of perihelion *)
                    N : real;    (* longitude of ascending node *)
                    a : real;    (* semi-major axis, in AU  *)
                    e : real;    (* eccentricity *)
                    Mo: real;    (* mean ancmoly at To,
                                           Mo = 0 at perihelion *)
               end;
var
     AsteroidFile: text;
     FileName: string[14];
     ElementSetRec: array [1..MaxRecSize] of ElementSet;
     Ifor: integer;
     MaxNoOfAsteroids: integer;
     Selected: integer;  (* record selected to work with *)
     A1,B1,Y1, A2,B2,Y2: real;   (* auxiliary quantities *)
     Px,Py,Pz, Qx,Qy,Qz: real;   (* Gaussian Constants *)
                     M : real;   (* Mean anomoly *)
                     no: real;   (* Mean daily motion,degrees per day *)
       EccentricAnomaly: real;   (* to solve Kepler's Equation *)
Xrefsun,Yrefsun,Zrefsun: real;   (* in AU  *)
                Rrefsun: real;   (* radius from sun, in AU  *)
         Xsun,Ysun,Zsun: real;   (* in AU  *)
Xrefgeo,Yrefgeo,Zrefgeo: real;   (* in AU  *)
                Rrefgeo: real;   (* sun to earth distance in AU  *)
                 RA,Dec: real;   (* right ascention, declination *)
             JulianDate: real;
           AsteroidName: name;
                     Tp: real;   (* date of perihelion passage/epoch *)
```

```
                         i : real;     (* inclination to ecliptic *)
                         w : real;     (* arguement of perihelion *)
                         N : real;     (* longitude of ascending node *)
                         a : real;     (* semi-major axis, in AU  *)
                         e : real;     (* eccentricity *)
                         Mo: real;     (* mean anomoly at To,
                                                  Mo = 0 at perihelion *)

function int(x:real): integer; (* int corresponds to BASIC  'int' *)
begin
  if (x >= 0) or (x = trunc(x)) then int := trunc(x)
             else  int := trunc(x) - 1
end; (* int *)


procedure GetDate(var Year,Mon,Day,Hour:real);

type
     reply = string[3];
     month = (Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec,
                                                 NotYetAssigned);

const      Blank = -1;  (* for dummy variables *)
           MinYear = 1900;  (*  limits of Julian Date formula *)
           MaxYear = 2099;  (*  limits of Julian Date formula *)
var
        WhatMonth: reply;
        MaxDays: integer;
        StartMonth, EndMonth:month;
        StartDay,EndDay,StartHour,EndHour:integer;
        GetYear, GetDay: integer;
        GetMonth: month;
        GetHour: real;
        Days,PhaseDays: real;
begin

      GetYear := Blank;
        while not (GetYear in [MinYear..MaxYear]) do
          begin
            write('What year, ',MinYear,' ... ',MaxYear,' ?  ');
            readln(GetYear);
          end; (* while *)
      Year := GetYear;

        while GetMonth = NotYetAssigned do
          begin
            write('What month, Jan..Dec ?       ');
            readln(WhatMonth);
              if WhatMonth = 'Jan' then GetMonth := Jan;
              if WhatMonth = 'Feb' then Getmonth := Feb;
              if WhatMonth = 'Mar' then Getmonth := Mar;
              if WhatMonth = 'Apr' then Getmonth := Apr;
```

```
                if WhatMonth = 'May' then GetMonth := May;
                if Whatmonth = 'Jun' then GetMonth := Jun;
                if WhatMonth = 'Jul' then GetMonth := Jul;
                if WhatMonth = 'Aug' then GetMonth := Aug;
                if WhatMonth = 'Sep' then GetMonth := Sep;
                if WhatMonth = 'Oct' then GetMonth := Oct;
                if WhatMonth = 'Nov' then GetMonth := Nov;
                if WhatMonth = 'Dec' then Getmonth := Dec
              end; (* while *)
         case GetMonth of
           Jan,Mar,May,Jul,Aug,Oct,Dec: MaxDays := 31;
           Sep,Apr,Jun,Nov: MaxDays := 30;
           Feb: MaxDays := 29
         end; (* case *)

         case GetMonth of
           Jan: Mon :=  1.0;
           Feb: Mon :=  2.0;
           Mar: Mon :=  3.0;
           Apr: Mon :=  4.0;
           May: Mon :=  5.0;
           Jun: Mon :=  6.0;
           Jul: Mon :=  7.0;
           Aug: Mon :=  8.0;
           Sep: Mon :=  9.0;
           Oct: Mon := 10.0;
           Nov: Mon := 11.0;
           Dec: Mon := 12.0;
         end; (* case *)

      GetDay := Blank;
         while not (GetDay in [1..MaxDays]) do
           begin
             write('What day, 1..',MaxDays,' ?              ');
             readln(GetDay)
           end; (* while *)
      Day := GetDay;

      GetHour := Blank;
         while not ((GetHour >= 0 ) and (GetHour <= 24.0) ) do
           begin
             write('What hour, 0.0  ... 24.0 ?    ');
             readln(GetHour)
           end; (* while *)
      Hour := GetHour;
end;  (* GetDate *)


procedure CalendarDate(JulianDate:real; var Day,Month,Year:integer);
                      (* works for dates 1100-2000 *)
                      (* appears slightly inaccurate before 1583 *)
label 1,2 ;
```

```
var
  NDate: real ;      (* temp *)
  Year1: real ;      (* temp *)
  LeapYear:integer ;
  NDay: real ;       (* temp *)

begin (* CalendarDate *)

  JulianDate := JulianDate -2400000.0 ;
  NDate := JulianDate - 15018.0 +0.0001 ; (* .0001 for roundoff error *)
  Year1 := NDate/365.25 ;
  Year := 1900 + int(Year1) ;

 if (Year/4-int(Year/4)=0 ) and (Year/100-int(year/100)<>0 ) then
                                                     LeapYear := 1;
 NDay := 365.25 *(Year1 - int(Year1) ) ;
 if NDay - int(NDay)> 0.5 then NDay := NDay + 1.0 ;
 Day := int(NDay) ;
 if LeapYear = 0 then Day := Day - 1 ;
 Day := Day + int((Year-2000)/100) ;
 if Year < 1583 then Day := Day - (10+int((Year - 1500) / 100)) ;
 if Day -  32 < 0 then begin
                         Month := 1 ;
                         Day := Day - 0  ;
                         goto 1 ;
                       end;
 if Day -  60 < 0 then begin
                         Month := 2 ;
                         Day := Day - 31 ;
                         goto 1 ;
                       end;
 if Day -  91 < 0 then begin
                         Month := 3 ;
                         Day := Day - 59 ;
                         goto 1 ;
                       end;
 if Day - 121 < 0 then begin
                         Month := 4 ;
                         Day := Day - 90 ;
                         goto 1 ;
                       end;
 if Day - 152 < 0 then begin
                         Month := 5 ;
                         Day := Day - 120 ;
                         goto 1 ;
                       end;
 if Day - 182 < 0 then begin
                         Month := 6 ;
                         Day := Day - 151 ;
                         goto 1 ;
                       end;
 if Day - 213 < 0 then begin
```

```
                        Month := 7 ;
                        Day := Day - 181 ;
                        goto 1 ;
                      end;
if Day - 244 < 0 then begin
                        Month := 8 ;
                        Day := Day - 212 ;
                        goto 1 ;
                      end;
if Day - 274 < 0 then begin
                        Month := 9 ;
                        Day := Day - 243 ;
                        goto 1 ;
                      end;
if Day - 305 < 0 then begin
                        Month := 10 ;
                        Day := Day - 273 ;
                        goto 1 ;
                      end;
if Day - 335 < 0 then begin
                        Month := 11 ;
                        Day := Day - 304 ;
                        goto 1 ;
                      end;
if Day - 366 < 0 then begin
                        Month := 12 ;
                        Day := Day - 334 ;
                      end;
1:if (LeapYear = 1) and (Month > 2) then Day := Day - 1 ;
 if Day >= 1 then goto 2 ;
 if (Day < 1) and (LeapYear = 1) and (Month = 3) then begin
                                                        Day := 29 ;
                                                        Month := 2 ;
                                                        goto 2 ;
                                                      end;
 if (Day < 1) and (LeapYear = 0) and (Month = 3) then begin
                                                        Day := 28 ;
                                                        Month := 2 ;
                                                        goto 2 ;
                                                      end;
 if (day < 1) and (Month = 1) then begin
                                     Day := 31 ;
                                     Month := 12 ;
                                     Year := Year - 1 ;
                                     goto 2 ;
                                   end;
 if (day < 1) and ((Month = 2) or (Month = 4) or (Month = 6)
          or (Month = 9) or (Month = 11)) then begin
                                                 Day := 31 ;
                                                 Month := Month - 1;
                                                 goto 2 ;
                                                  end;
```

```
 if Day < 1 then begin
                  Day := 30 ;
                  Month := Month - 1 ;
                end;
2: if Day = 365 then begin
                      Day := 31 ;
                      Month := 12 ;
                    end;
   if Day = 366 then begin
                      Day := 1 ;
                      Month := 1 ;
                      Year := Year + 1 ;
                    end;

end; (* CalendarDate *)


procedure GetJulianDate (Year,Month,Day,UnivTime:real;
                                                var JulianDate:real);

  (* input: Universal Time and Date from 28 Feb 1900 to 31 Dec 2099 *)

  (* output: Julian Date  2 400 000 . 0000  *)

begin (* GetJulianDate *)

JulianDate := 367 * Year - trunc(7/4 * (Year + trunc((Month + 9) / 12)))
                         + trunc(275 * Month / 9) + Day + 1721013.5
                         + UnivTime / 24 ;

end; (* GetJulianDate *)



procedure SolveKeplersEqn (M:real; var EccentricAnomaly:real);
  const
    accuracy = 0.0000000001;

  var
    Mtemp: real;
    Mrad: real;
    Iteration: real;
    diff: real;  (* temporary difference *)
    EccentricAnomalyRadians: real;

begin
  Mtemp := -1.0;  (* artificial starting value *)
  Mrad := M*Pi/180.0;
  EccentricAnomalyRadians := Mrad - 0.1 ;  (* rough starting point *)

 while Mtemp < Mrad do
```

```
    begin
      EccentricAnomalyRadians := EccentricAnomalyRadians + 0.1;
      Mtemp := EccentricAnomalyRadians - e*sin(EccentricAnomalyRadians);
    end;

  while Mtemp > Mrad do
    begin
      EccentricAnomalyRadians := EccentricAnomalyRadians - 0.1;
      Mtemp := EccentricAnomalyRadians - e*sin(EccentricAnomalyRadians);
    end;

    iteration := 0.1;
    diff := 1.0; (* artificial start *)
    while diff > accuracy do
      begin
        iteration := iteration/2.0;
        if Mtemp < Mrad
         then
          EccentricAnomalyRadians := EccentricAnomalyRadians + iteration
         else
          EccentricAnomalyRadians := EccentricAnomalyRadians - iteration;

     Mtemp := EccentricAnomalyRadians - e*sin(EccentricAnomalyRadians);
     diff := abs(Mrad - Mtemp);
     end; (* while *)

     EccentricAnomaly := EccentricAnomalyRadians * 180.0/Pi;

end; (* procedure SolveKeplersEqn *)



procedure SunXYZcoord(JulianDate: real; var Xcoord: real;
                                        var Ycoord: real;
                                        var Zcoord: real);
var
   L: real;
   g: real;
   gamma: real;
   E: real;
   R: real;
   n: real;

begin
   n := nAdjust1985 + JulianDate - JD00Jan1985;
   g := 357.528 + 0.9856003 * n;

     while g < 0.0
       do g := g + 360.0 ;
     while g > 360.0
       do g := g - 360.0 ;
   L := 280.460 + 0.9856474 * n ;
```

```
      while L < 0.0
        do L := L + 360.0 ;
      while L > 360.0
        do L := L - 360.0 ;

 gamma := L + 1.915 * sin(g*Pi/180.0) + 0.020 * sin(2.0 * g * Pi/180.0);

 E := 23.439 - 0.0000004 * n ;

 R := 1.00014 - 0.01671 * cos(g * Pi/180.0) - 0.00014 * cos(2.0 * g
                                                      * Pi/180.0);

    Xcoord := R * cos(gamma * Pi/180.0) ;
    Ycoord := R * cos(E * Pi/180.0) * sin(gamma * Pi/180.0) ;
    Zcoord := R * sin(E * Pi/180.0) * sin(gamma * Pi/180.0) ;


end ; (*  SunXYZcoord *)



procedure GetDataFile;  (* Reads data into global variables *)

     (* *** Same as procedure GetDataFile ********** *)
     (* *** in program HulkowerLauBenderDeltaVee *** *)


procedure SelectObject; (* Global variables *)

     (* *** Same as procedure SelectObject ********* *)
     (* *** in program HulkowerLauBenderDeltaVee *** *)


procedure CalculateObjPosition (DateToCalculate: real);
begin
  SunXYZcoord(DateToCalculate,Xsun,Ysun,Zsun);
  EccentricAnomaly := 0.0;
  A1 := sin(N*Pi/180.0)*sin(w*Pi/180.0);    (* auxiliary quantities *)
  B1 := cos(N*Pi/180.0)*sin(w*Pi/180.0);
  Y1 := sin(i*Pi/180.0)*sin(w*Pi/180.0);

  A2 := sin(N*Pi/180.0)*cos(w*Pi/180.0);
  B2 := cos(N*Pi/180.0)*cos(w*Pi/180.0);
  Y2 := sin(i*Pi/180.0)*cos(w*Pi/180.0);

                                            (* GAUSSIAN CONSTANTS *)
Px := B2 - A1*cos(i*Pi/180.0);
Py := (A2 + B1*cos(i*Pi/180.0))*cos(ObliqEclip*Pi/180.0) -
                                         Y1*sin(ObliqEclip*Pi/180.0);
Pz := (A2 + B1*cos(i*Pi/180.0))*sin(ObliqEclip*Pi/180.0) +
                                         Y1*cos(ObliqEclip*Pi/180.0);
```

```
Qx := -B1 - A2*cos(i*Pi/180.0);
Qy := (-A1 + B2*cos(i*Pi/180.0))*cos(ObliqEclip*Pi/180.0) -
                                        Y2*sin(ObliqEclip*Pi/180.0);
Qz := (-A1 + B2*cos(i*Pi/180.0))*sin(ObliqEclip*Pi/180.0) +
                                        Y2*cos(ObliqEclip*Pi/180.0);


no := 0.9856076686/sqrt(a*a*a);  (* mean daily motion in degrees *)
M := Mo + no*(DateToCalculate - Tp); (* mean anomaly, DateToCalculate is
                                                time of ephemeris *)

SolveKeplersEqn (M, EccentricAnomaly);

Xrefsun := a*Px*(cos(EccentricAnomaly*Pi/180.0)-e)+
                a*sqrt(1-e*e)*Qx*sin(EccentricAnomaly*Pi/180.0);
Yrefsun := a*Py*(cos(EccentricAnomaly*Pi/180.0)-e)+
                a*sqrt(1-e*e)*Qy*sin(EccentricAnomaly*Pi/180.0);
Zrefsun := a*Pz*(cos(EccentricAnomaly*Pi/180.0)-e)+
                a*sqrt(1-e*e)*Qz*sin(EccentricAnomaly*Pi/180.0);

Rrefsun := sqrt(Xrefsun*Xrefsun + Yrefsun*Yrefsun + Zrefsun*Zrefsun);
                                                        (* in AU *)


   Xrefgeo := Xrefsun + Xsun ;
   Yrefgeo := Yrefsun + Ysun ;
   Zrefgeo := Zrefsun + Zsun ;

Rrefgeo := sqrt(Xrefgeo*Xrefgeo + Yrefgeo*Yrefgeo + Zrefgeo*Zrefgeo);
                                                        (* in AU *)

RA := arctan(Yrefgeo/Xrefgeo)*180.0/Pi;   (* RA in degrees *)

if Xrefgeo<0 then RA := RA + 180.0
      else if Yrefgeo<0 then RA := RA + 360.0 ; (* correct quadrant *)
RA := RA/15.0; (* RA in hours *)
DEC := arctan((Zrefgeo/Rrefgeo)/sqrt(1.0 - Zrefgeo/Rrefgeo*Zrefgeo
                        /Rrefgeo)) * 180.0/Pi; (* in degrees *)
end; (* CalculateObjPosition *)



procedure InitialDisplay;

     (* *** Same as procedure InitialDisplay ******* *)
     (* *** in program HulkowerLauBenderDeltaVee *** *)


procedure PeriodEphemeris;
var
  Year,Month,Day,Hour: real;
  EphemerisInterval: real;
```

```
  DateToCalculate: real;
  StartJD,EndJD: real;
  Choice: char;
  PrinterOrScreen: char;
  RAhour, RAmin, DECdeg, DECmin, DECsec: integer;
  RAsec: real;


procedure Print(DateToPrint: real);
   var
     Day,Month,year: integer;

   begin (* Print *)

     CalendarDate(DateToPrint,Day,Month,Year);

     if PrinterOrScreen in ['S','s']
       then
         begin
           write(Day:2,Month:3,Year:5,' ....');
           writeln(RAhour:3,RAmin:3,RAsec:5:1,DECdeg:6,
                                             DECmin:3,DECsec:3 );
         end (* then *)
       else
         begin
           write(LST,Day:2,Month:3,Year:5,' ....');
           writeln(LST,RAhour:3,RAmin:3,RAsec:5:1,DECdeg:6,
                                             DECmin:3,DECsec:3 );
         end; (* else *)
      end; (* Print *)


procedure CalcEphemeris;
           procedure TruncateRADEC;
              begin
                RAhour := trunc(RA);
                RAmin := trunc(frac(RA)*60);
                RAsec := frac(frac(RA)*60)*60;

                DECdeg := trunc(DEC);
                DECmin := abs(trunc(frac(DEC)*60));
                DECsec := abs(trunc(frac(frac(DEC)*60)*60));
              end;(* TruncateRADEC *)

      begin (* CalcEphemeris *)
         DateToCalculate :=  StartJD;

        while DateToCalculate < EndJD do
             begin
                CalculateObjPosition(DateToCalculate);
                TruncateRADEC;
```

```
                Print(DateToCalculate);
                DateToCalculate := DateToCalculate + EphemerisInterval;
             end; (* while *)

                DateToCalculate := EndJD;
                CalculateObjPosition(DateToCalculate);
                TruncateRADEC;
                Print(DateToCalculate);

        end; (* CalcEphemeris *)


    begin (* PeriodEphemeris *)
      writeln;
      writeln('Starting Date');
      GetDate(Year,Month,Day,Hour);
      GetJulianDate(Year,Month,Day,Hour,JulianDate);
      StartJD := JulianDate;
      repeat
        writeln;
        writeln('For a SINGLE position,  type  <S>');
        writeln('For MULTIPLE positions, type  <M>');
        readln(Choice)
      until Choice in ['S','s','M','m'];

      if Choice in ['S','s'] then
          begin
            EndJD := StartJD;
            EphemerisInterval := 1.0; (* Dummy *)
          end
                        else
          begin
            writeln;
            writeln('End Date');
            GetDate(Year,Month,Day,Hour);
            GetJulianDate(Year,Month,Day,Hour,JulianDate);
            EndJD := JulianDate;
              writeln;
              write('What is interval for ephemeris in decimal days? ');
              readln(EphemerisInterval);
          end; (* else *)

      writeln;
      repeat
         writeln('Output to   <S> Screen   or   <P> Printer    ??');
         readln(PrinterOrScreen);
      until PrinterOrScreen in ['S','s','P','p'];

      if PrinterOrScreen in ['S','s'] then
          begin
            writeln;
```

```
      writeln('Ephemeris for   >>> ',AsteroidName,' <<<');
      writeln('DD MM YEAR      Right Ascen  Declination');
   end
                                     else
   begin
      writeln(LST);
      writeln(LST,'Ephemeris for   >>> ',AsteroidName,' <<<');
      writeln(LST,'DD MM YEAR      Right Ascen  Declination');
   end;

  CalcEphemeris;

end; (* PeriodEphemeris *)


begin (* main *)
  InitialDisplay;
  GetDataFile;
  SelectObject;
  PeriodEphemeris;
  writeln;
  writeln;
  readln; (* return to DOS *)
end. (* main *)
```

Appendix D.

PASCAL Program to Generate True Anomalies verses Date

```
program CalcAnomalies;

(* Calcs True Anomalies every 10 days from 0h - 05 Jan 1985 to 0h      *)
(* - 2 Jan 2020 for Earth and Target. Stores values in Vearth[0..1278]*)
(* and Vtarget[0..1278]. Saves them in user-designated file           *)
(* Note: element set must be entered directly in code. Program does   *)
(* not read .ELE files.                                                *)

const
  usun   = 1.32718E20;    (* Grav const of sun in m3/s2 *)
    AU   = 1.49600E11;    (* Astronaumical Unit in m *)
var
  Mearth  : real;   (* Mean anomaly earth at epoch, in degrees *)
  Tearth  : real;   (* Time of epocn earth  *)
  eearth  : real;   (* Eccentricity earth  *)
  aearth  : real;   (* Semi-major axis a of earth in m *)
  Vearth  : array[0..2000] of real; (* True anomaly of earth *)

  Mtarget  : real;   (* Mean anomaly target at epoch, in degrees *)
  Ttarget  : real;   (* Time of epoch target  *)
  etarget  : real;   (* Eccentricity target  *)
  atarget  : real;   (* Semi-major axis a of target in m *)
  Vtarget  : array[0..2000] of real; (* True anomaly of target *)

   counter  : integer;
        JD  : real;   (* Julian date *)
      Name  : string[8];


function tan(x:real):real;
  begin
    tan := sin(x)/cos(x);
  end;


procedure KeplerEquation(M,e:real; var X:real);

(* Adapted from :
 Danby, J.M.A and T.M. Burkardt. "The Solution of Kepler's Equation, I,"
 Celestial Mechanics, 31: 95-107 (May 1983).   *)

var
       ES: real;
        F: real;
       EC: real;
       FP: real;
       DX: real;
```

```
begin
   write('*');
    X := M;            (* Initial Guess *)
   ES := e * sin(X);
    F := X - ES - M;

  repeat
   EC := e * cos(X);
   FP := 1.0 - EC;
   DX := -F/FP;
   DX := -F/(FP + DX*ES/2.0);
   DX := -F/(FP + DX*ES/2.0 + DX*DX*EC/6.0);
   DX := - F / (FP + DX * ES/2.0 + DX * DX * EC/6.0
                              - DX * DX * DX * ES/24.0);
   X := X + DX;
   ES := e * sin(X);
    F := X - ES - M;
  until ABS(f) <= 0.0000000001;

end;


procedure Bacon(M,T,e,a: real; z: integer);

var
  counter : integer;

  tp: real;   (* Time since perihelion passage  *)
  MX: real;   (* Calculated mean anomaly  *)
  X : real;   (* Eccentric anomaly  *)
  V : real;   (* True anomaly  *)
 Tpo: real;   (* Time of perihelion passage  *)
  n : real;   (* Mean motion  *)
     JD : real;   (* Current Julian date for calculations *)

begin

  n := sqrt (  usun /  (a*a*a)  ) * 3600.0 * 24.0 ;
  tp := M / n ;
  Tpo := T - tp;
  JD := 2446066.5;          (*  0h - 01 Jan 1985 *)
  JD := 2446066.5 + 4.0;    (* Start on mult of 10 (2446070.5),
                                        0h - 05 Jan 1985 *)
  counter := 0;

  while Tpo > JD do
   begin
     Tpo := Tpo - 2.0 * Pi / n;
   end;

  while JD < 2458850.5 do     (*  0h - 02 Jan 2020  *)
   begin
```

```
    MX := n * (JD - Tpo);

    while MX >= (2.0 * Pi) do
      begin
        MX := MX - 2.0 * Pi;
        Tpo := Tpo + 2.0 * Pi / n;
      end;

  KeplerEquation(MX,e,X);

  V := 2.0 * arctan(tan(0.5 * X) *
                   sqrt( (1.0 + e)/(1.0 - e) ) );

  if V < 0.0 then V := V + 2.0 * Pi ;
   V := V*180.0/Pi ;
  if z = 1 then Vearth[counter] := V else Vtarget[counter] := V;
  JD := JD + 10.0;

  counter := counter + 1;

 end; (* while JD < 244... *)
end; (* procedure Bacon *)


procedure SaveAnomalyFile;
const
  MaxNoAnomalies = 1278 ;
type
  AnomRec = record
              Ve : real;
              Vt : real;
             end;
var
  AnomFile : file of AnomRec;
  Anom     : AnomRec;
  counter  : integer;
begin
  writeln;
  writeln('Input prefix for anomaly data file  ');
  readln(Name);
  assign(AnomFile,Name+'.DTA');
  Rewrite(AnomFile);
  with Anom do
    begin
      for counter := 0 to MaxNoAnomalies do
        begin
          Ve := Vearth[counter];
          Vt := Vtarget[counter];
         write(AnomFile,Anom);
        end;
    end; (* with Anom *)
```

```
    close(AnomFile);
end; (* Procedure SaveAnomalyFile *)


procedure LoadAnomalyFile;
const
  MaxNoAnomalies = 1278 ;
type
  AnomRec = record
              Ve : real;
              Vt : real;
             end;
var
  AnomFile : file of AnomRec;
  Anom     : AnomRec;
  counter  : integer;
begin
  assign(AnomFile,'ANOM-1.DTA');
  Reset(AnomFile);
  with Anom do
    begin
      for counter := 0 to MaxNoAnomalies do
        begin
          read(AnomFile,Anom);
          Vearth[counter] := Ve;
          Vtarget[counter] := Vt;
        end;
    end; (* with Anom *)
    close(AnomFile);
end; (* Procedure LoadAnomalyFile *)



begin
  Mearth := 208.89818 * Pi/180.0;
  Tearth := 2446280.5;     (* All 1985 Astronomical Almanac data *)
  eearth := 0.0166912;
  aearth := 1.4960328E11;

  bacon(Mearth, Tearth, eearth,aearth,1);

   (*  ** Asteroid 1982 XB **      Hulkower, Lau, Bender data *)

  Mtarget := 266.0197 * Pi/180.0;
  Ttarget := 2446000.5;
  etarget := 0.4468762;
  atarget := 1.837667 * AU;

  bacon(Mtarget, Ttarget, etarget,atarget,0);

  JD := 2446070.5;
  writeln;
```

```
(* Remove brlaces here for simultaneous listing *)
(*   for counter := 0 to 1278 do  *)
                          (* 2446070.5 + 10.0 * 1278 = 2458850.5 *)
(*     begin
      writeln(JD + 10.0*counter:12:1,Vearth[counter]:8:1,
                                 Vtarget[counter]:8:1);
    end;          *)

SaveANomalyFile;

end.
```

Appendix E.

PASCAL Program to List True Anomalies verses Date

```
program ListAnomalies;

(* List True Anom every 10 days  1985-2020. 05 Jan 1985 - 02 Jan 2020 *)
(* Locks for file ANOM-1.DTA. Lists True Anomalies for both departure *)
(* and arrival planets in separate tabular form.                      *)

var
   Vearth, Vtarget, V: array[0..1278] of real;


procedure LoadAnomalyFile;
const
  MaxNoAnomalies = 1278 ;

type
  AnomRec = record
              Ve : real;
              Vt : real;
            end;
var
  AnomFile : file of AnomRec;
  Anom     : AnomRec;
  counter  : integer;

begin
  assign(AnomFile,'ANOM-1.DTA');
  Reset(AnomFile);
  with Anom do
    begin
      for counter := 0 to MaxNoAnomalies do
        begin
          read(AnomFile,Anom);
          Vearth[counter] := Ve;
          Vtarget[counter] := Vt;
        end;
    end; (* with Anom *)
    close(AnomFile);
end; (* Procedure LoadAnomalyFile *)


procedure ListAnom;
const
  JD = 2446070.5;
var
  counter, column : integer;
begin

  for counter := 0 to 127 do   (* 2446070.5 + 100.0 * 127 = 2458770.5 *)
```

```
    begin

     if ( counter = 0) or (counter = 43) or (counter = 86)  then
     begin
     readln;
     writeln(LST); writeln(LST);
     writeln(LST); writeln(LST); writeln(LST);

     writeln(LST,
      '           Julian Date   70  80  90  00  10  20  30  40  50  60');
     writeln(LST,
      '           -----------  --- --- --- --- --- --- --- --- --- ---');
     end;

      write(LST,'        ',JD + 100.0*counter:12:1,'  ');
        for column := 0 to 9 do
          begin
            write(LST,V[counter+column]:4:0);
          end;
        writeln(LST);
    end;
end; (* procedure ListAnom *)


begin

  LoadAnomalyFile;
  V := Vearth;
  ListAnom;
  V := Vtarget;
  ListAnom;

end.
```

Appendix F.

PASCAL Program to Find Launch Dates

```
program FindLaunchDate;

(* Finds a match of 2 true anomalies and time-of-flight if a match    *)
(* exists from January 1985 to January 2020. Resolution is 10 degrees *)
(* of true anomaly and 10 day intervals of dates. Each search is for  *)
(* true anomaly of departure plus/minus 10 degrees, true anomaly of   *)
(* arrival plus/minus 10 degrees, and time-of-flight plus or minus 10 *)
(* days. A total of 27 combinations are searched (3x3x3) for each set *)
(* of true anomalies and TOF entered. Uses data file ANOM-1.DTA or    *)
(* user supplied data file. This data file contains 1278 entries of   *)
(* true anomalies every 10 days from Jan 1985 to Jan 2020 for both    *)
(* departure and arrival planets.                                     *)

const
  MaxNoAnomalies = 1278 ;
  MaxArraySize   = 1279 ;

var
      LaunchFile : text;
  LaunchFileName : string[12];
          DeltaV : real;
               M : string[9];
             D,Y : integer;

    TAEarth : array[1..MaxArraySize] of integer;
   TATarget : array[1..MaxArraySize] of integer;
    TAE,TAT, ITAEarth, ITATarget : integer;
                 TOF, TOF1, TOF2 : integer;
                             Agn : char ;
                           Again : boolean;

procedure LoadAnomalyFile;
type
  AnomRec = record
              Ve : real;
              Vt : real;
            end;
var
      AnomFile : file of AnomRec;
      Anom     : AnomRec;
      counter  : integer;
  AnomFileName : string[12];

begin
  writeln; writeln('Input prefix of Anomaly file  ');
  readln(AnomFileName);
  assign(AnomFile,AnomFileName+'.DTA');
  Reset(AnomFile);
```

```
  with Anom do
    begin
      for counter := 1 to MaxNoAnomalies do
        begin
          read(AnomFile,Anom);
          TAEarth[counter] := round(Ve/10.0);
          TATarget[counter] := round(Vt/10.0);
        end;
    end; (* with Anom *)
    close(AnomFile);
end; (* Procedure LoadAnomalyFile *)

procedure JulianToDate(JD:real);

(* Convert Julian dates to calendar dates.                               *)
(* Works from 1 Jan 1900 to near end of 21st century.                    *)
(* Writes date in form: e.g.  12 September 1984                          *)
(* Julian Date procedures adapted from public domain software: author *)
(* unknown                                                               *)

var date : real;
    Julian,Day,Month,Year: integer;



procedure JtoD(Julian: integer;var Day,Month,Year: integer);
  (*  Convert from a Julian date to a calendar date  *)
  (*  Note that much care is taken to avoid problems with inaccurate bit
   representations inherent in the binary fractions of the real numbers
   used as temporary variables.  Thus the seemingly unnecessary use of
   small fractional offsets and int() functions  *)

  var Temp: real;
  begin
  Temp := int(32767.5+Julian); (* Convert 16 bit quantity to real no. *)
  if Temp<58.5
   then
    begin        (*  The first two months of the twentieth century are
                  handled as a special case of the general algorithm
                  used which handles all of the rest  *)
    Year := 1900;
    if Temp<30.5
     then
      begin
      Month := 1;
      Day := round(Temp+1.0)
      end
     else
      begin
      Month := 2;
      Day := round(Temp-30.0)
      end
```

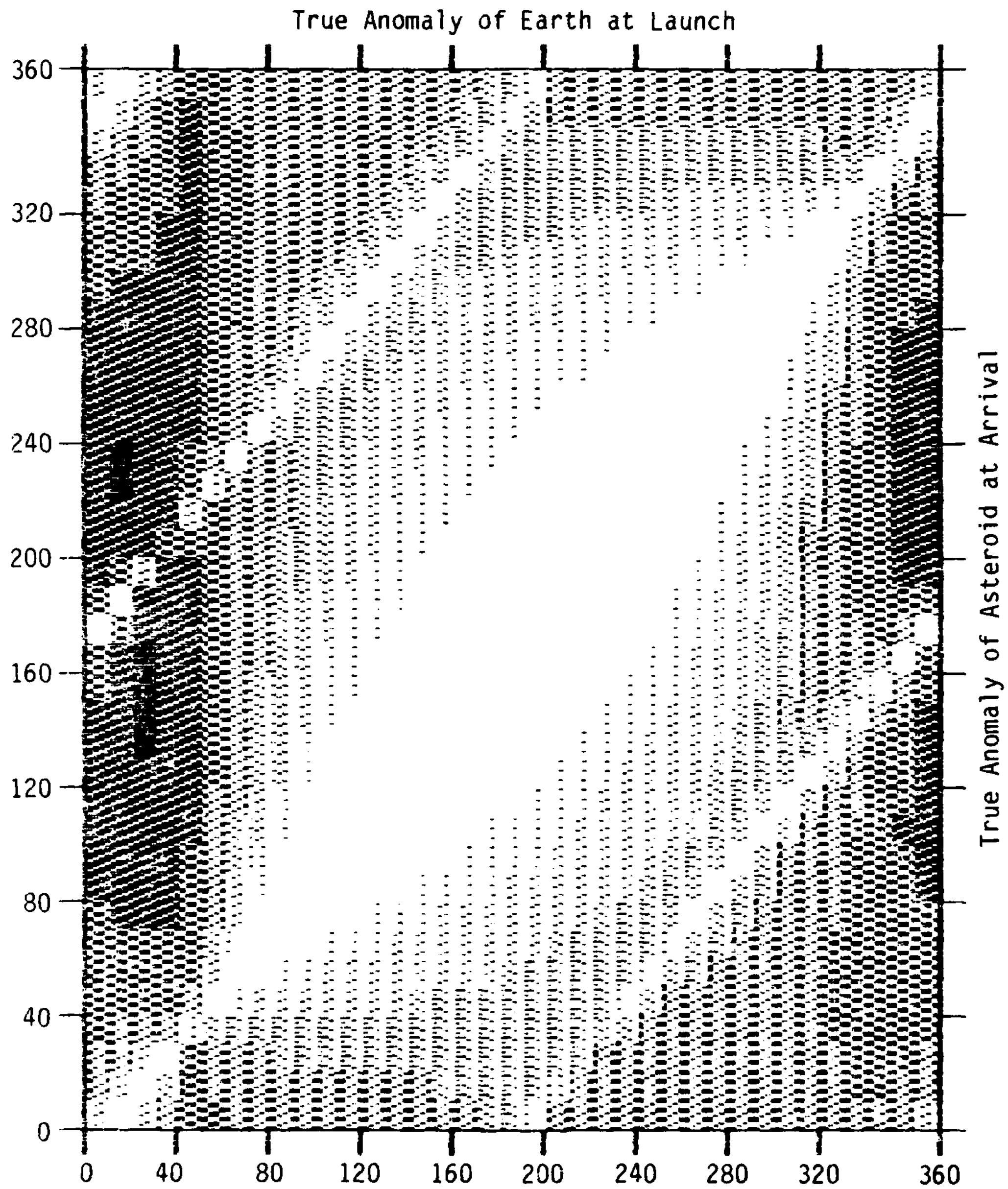
```
    end
   else
    begin
    Temp := int(4.0*(Temp-59.0)+3.5);
    Year := trunc(Temp/1461.0+0.00034223);
      (*  0.00034223 is about one half of the reciprocal of 1461.0  *)

    Day := succ(round(Temp-Year*1461.0) div 4);
    Month := (5*Day-3) div 153;
    Day := succ((5*Day-3) mod 153 div 5);
    Year := Year+1900;
    if Month<10
     then
      Month := Month+3
     else
      begin
      Month := Month-9;
      Year := succ(Year)
      end
    end
  end;


procedure WriteDate(Julian: integer);
  (*  Write the date out to the console in long form ,
                    e.g. "10 September 1984"  *)
  const
        Months: array[1..12] of string[9] =
           ('January','February','March','April','May','June',
            'July','August','September','October',
            'November','December');
  var Day,Month,Year : integer;

  begin
  JtoD(Julian,Day,Month,Year);         (*  Convert into date form  *)
  write(Day,' ',Months[Month],' ',Year);

        D := Day;
        M := Months[Month];
        Y := Year;

  end;


begin (* JulianToDate *)

  JD := JD - 2447787.99999;
  Julian := round(jd);
  WriteDate(Julian);

end; (* JulianToDate *)
```

```
procedure InputTAandTOF;

begin
  TATarget[1279] := 1000;  (* Dummy for search routine *)
   writeln;
   write('Input True Anomaly of Departure Planet in degrees  ');
   readln(ITAEarth);
   ITAEarth := round(ITAEarth/10);
   writeln;
   write('Input True Anomaly of Arrival Planet in degrees  ');
   readln(ITATarget);
   ITATarget := round(ITATarget/10);
   writeln;
   write('Input Delta V in km/sec   ');
   readln(DeltaV);
   writeln;
   write('Input Time-of-Flight in days   ');
   readln(TOF);
   writeln;


   TOF  := round(TOF/10); (* Time-of-Flight in 10*days *)
   TOF1 := TOF + 1;
   TOF2 := TOF - 1;
end; (* InputTAandTOF *)

procedure DoCombos;
var
  counter : integer;

  procedure CheckCombos;

  procedure WriteDetails(increment: integer);
  begin
    JulianToDate(2446070.5 + 10.0*counter);
    writeln(TAE*10:4,TAT*10:4,
              10*(TOF+increment):5, TOF:6, DeltaV:4:1);

      writeln(LaunchFile,D:2,' ',M,' ',Y:4,'   ',TAE*10:4,TAT*10:4,
              10*(TOF+increment):5, TOF:6, DeltaV:4:1);
  end; (* WriteDetails *)


  procedure CheckTATarget;
  begin
    if TATarget[counter+TOF]   = ITATarget then WriteDetails(0);
    if TATarget[counter+TOF+1] = ITATarget then WriteDetails(1);
    if TATarget[counter+TOF-1] = ITATarget then WriteDetails(-1);
  end; (* CheckTATarget *)

  begin  (* CheckCombos *)
    for TAE := ITAEarth-1 to ITAEarth+1 do
```

```
      begin
        if TAE = -1 then TAE := 35;
        if TAE = 37 then TAE := 10;
        for TAT := ITATarget-1 to ITATarget+1 do
          begin
            if TAT = -1 then TAT := 35;
            if TAT = 37 then TAT := 10;
            if TAEarth[counter] = ITAEarth then CheckTATarget;
          end; (* for TAT *)
      end; (* forTAE *)
  end; (* CheckCombos *)

begin (* DoCombos *)
  counter := 1;
  while counter < 1279-TOF do
   begin
     CheckCombos;
     counter := counter + 1;
   end; (* while counter *)
end; (* DoCombos *)


procedure InitLaunchFile;

begin
  writeln;
  writeln('Input prefix of Launch file  ');
  readln(LaunchFileName);

  assign(LaunchFile,LaunchFileName+'.LAU');
  Rewrite(LaunchFile);
end; (* Procedure InitLaunchFile *)



begin (* Main FindLaunchDate *)
  LoadAnomalyFile;
  InitLaunchFile;
  again := true;
  while Again = true do
    begin
      InputTAandTOF;
      DoCombos;
      writeln;
      writeln('Again ?  (Y/N)  ');
      readln(Agn);
      if upcase(agn) = 'N' then Again := false;
    end;
  close(LaunchFile);
  writeln('END END');
  readln;
end. (* Main FindLaunchDate *)
```

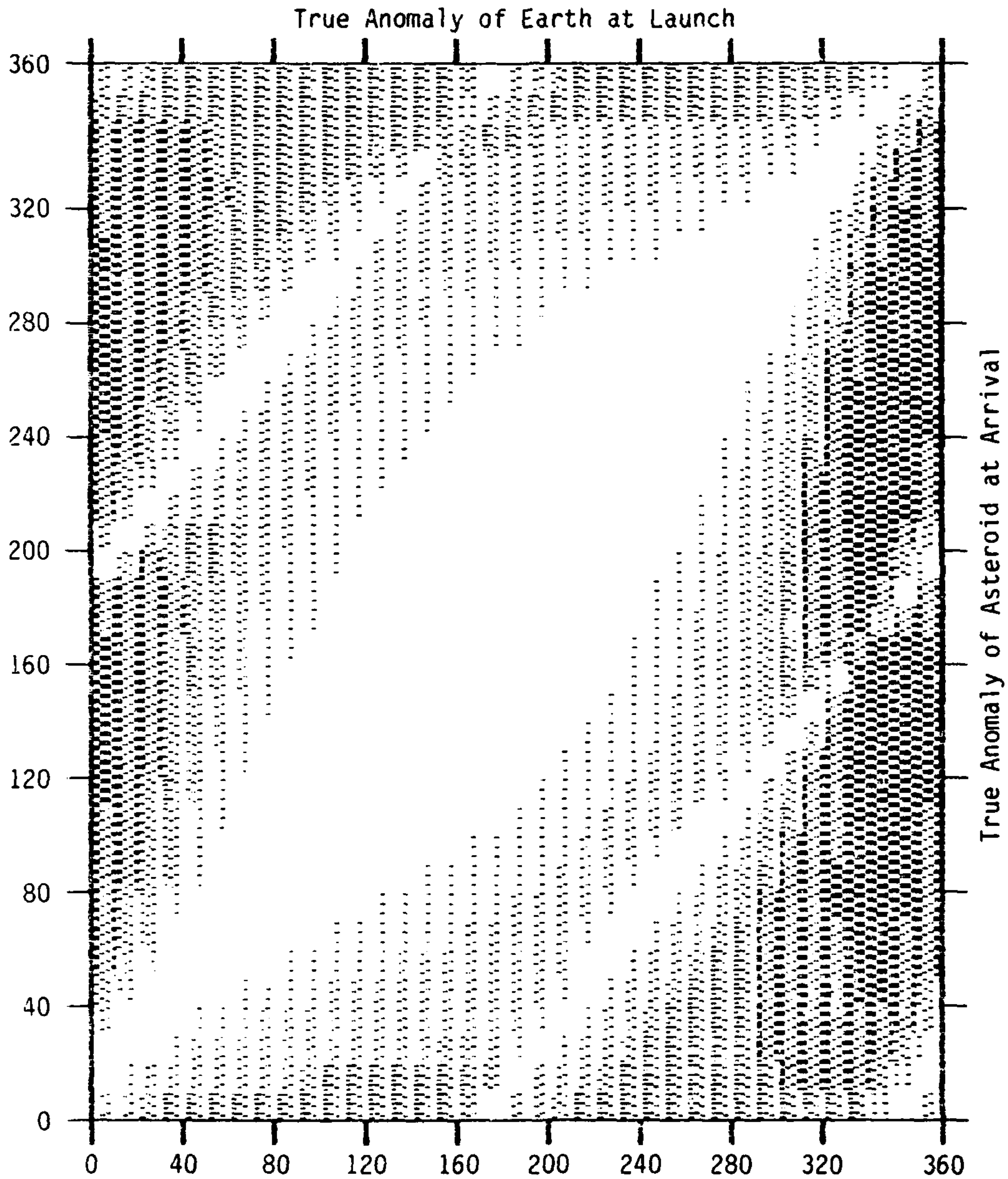
Appendix G. Prime Rib Plots for Six Selected Asteroids

ASTEROID 1982DB



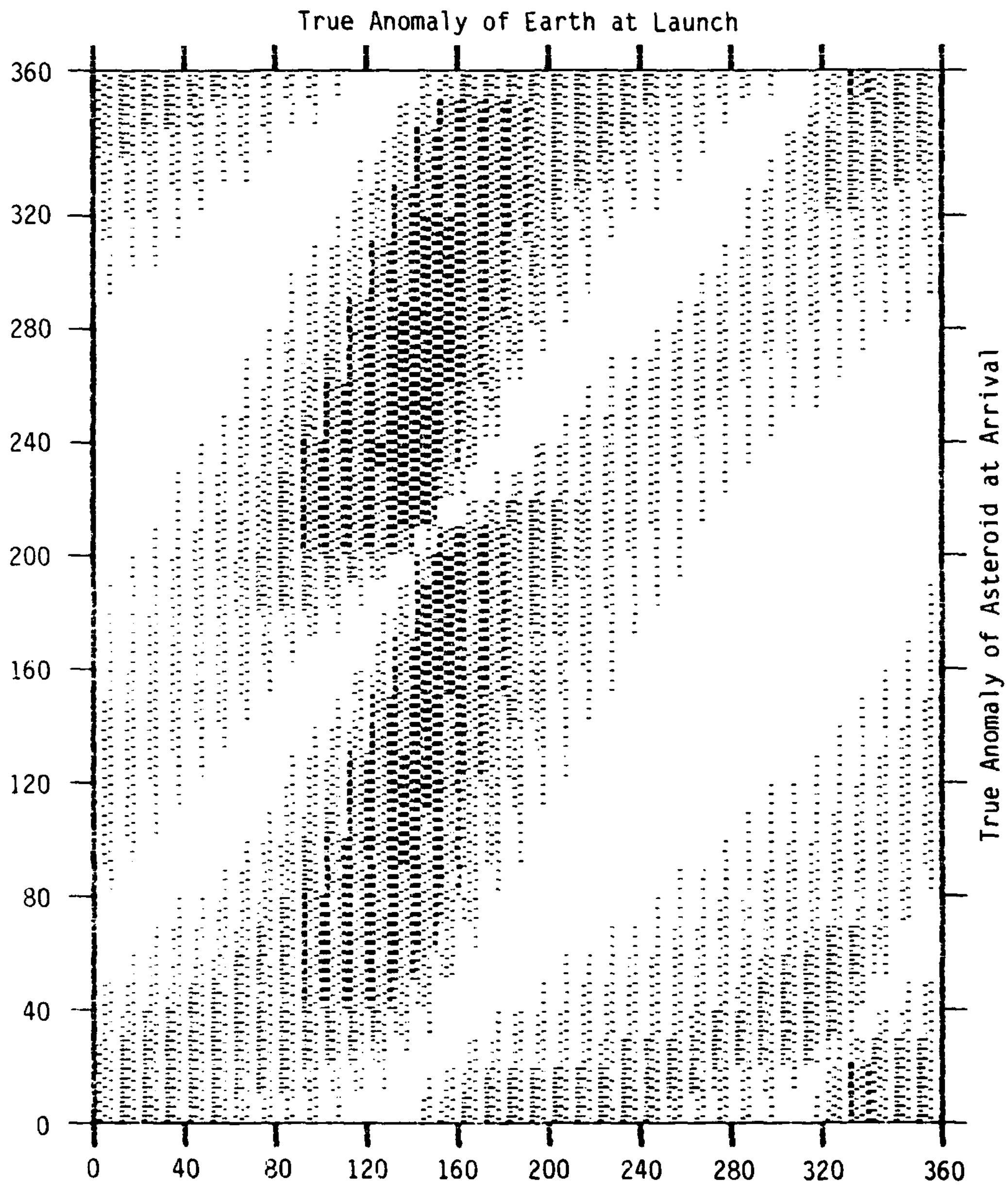
(Grey Scale: Black ≤ 4.5 km/s, White > 20 km/s)

ASTEROID 1982XB



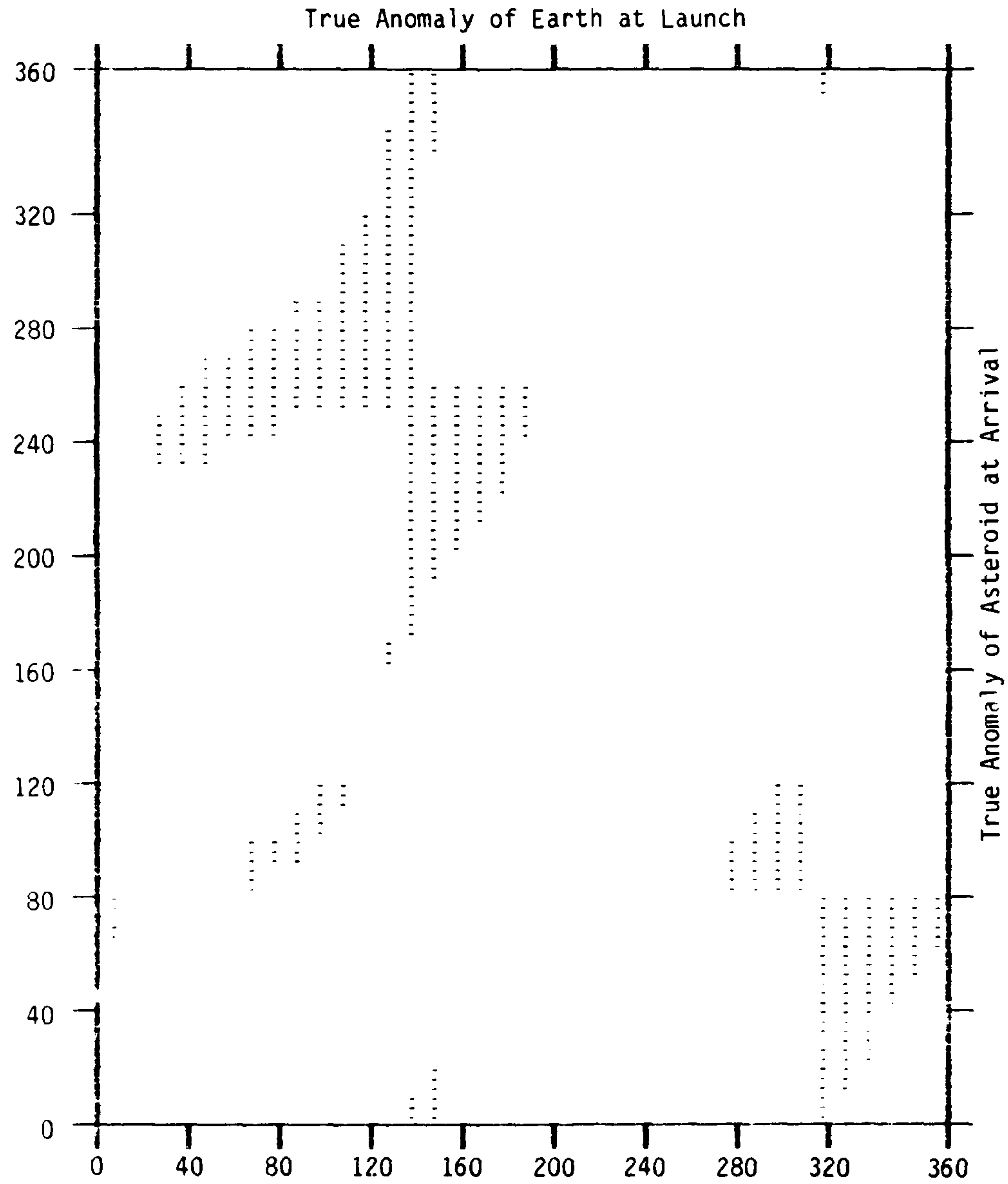
(Grey Scale: Black $\leq$ 4.5 km/s, White > 20 km/s)

ASTEROID ANTEROS



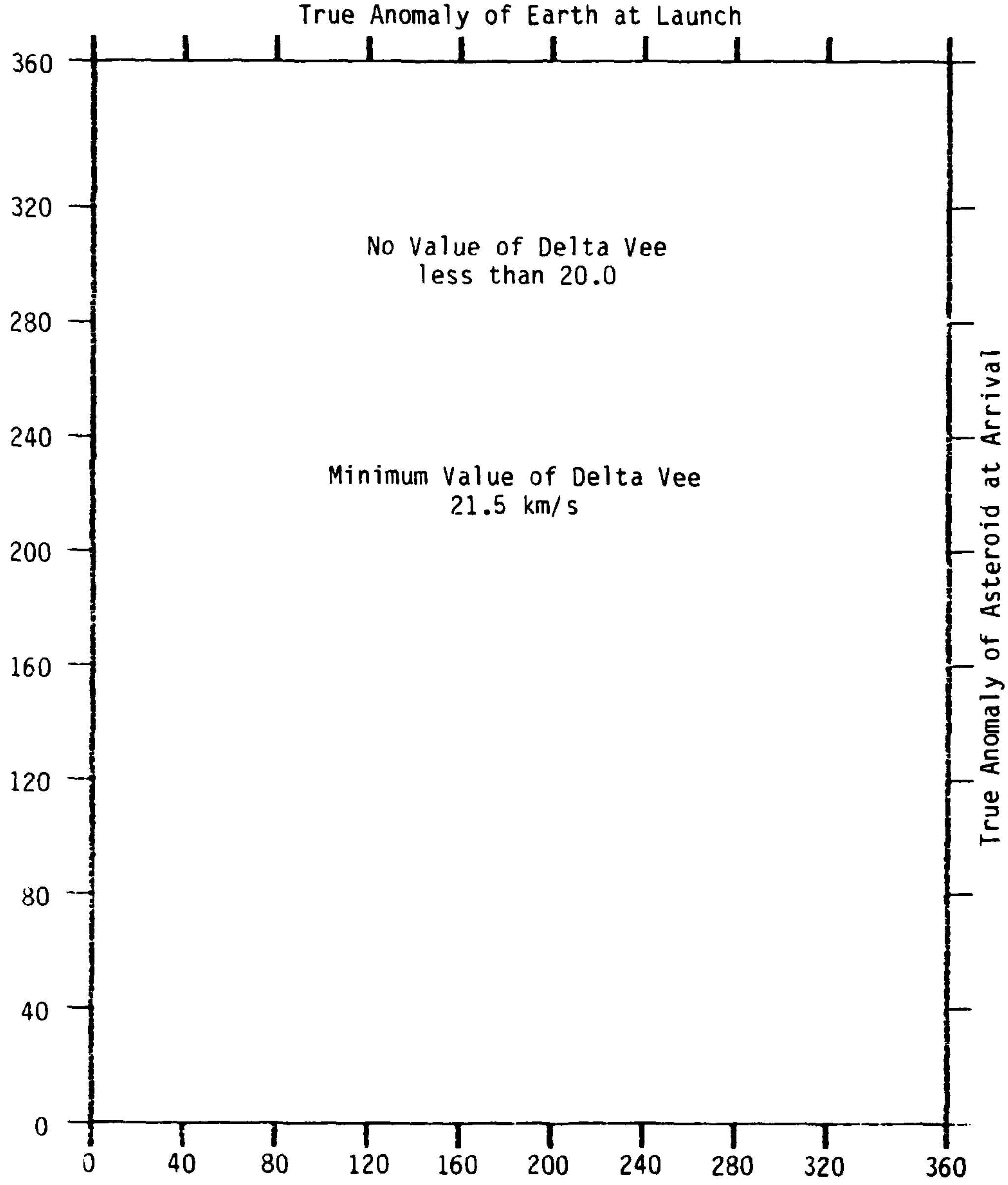
(Grey Scale: Black $\leq$ 4.5 km/s, White > 20 km/s)

ASTEROID 1985JA



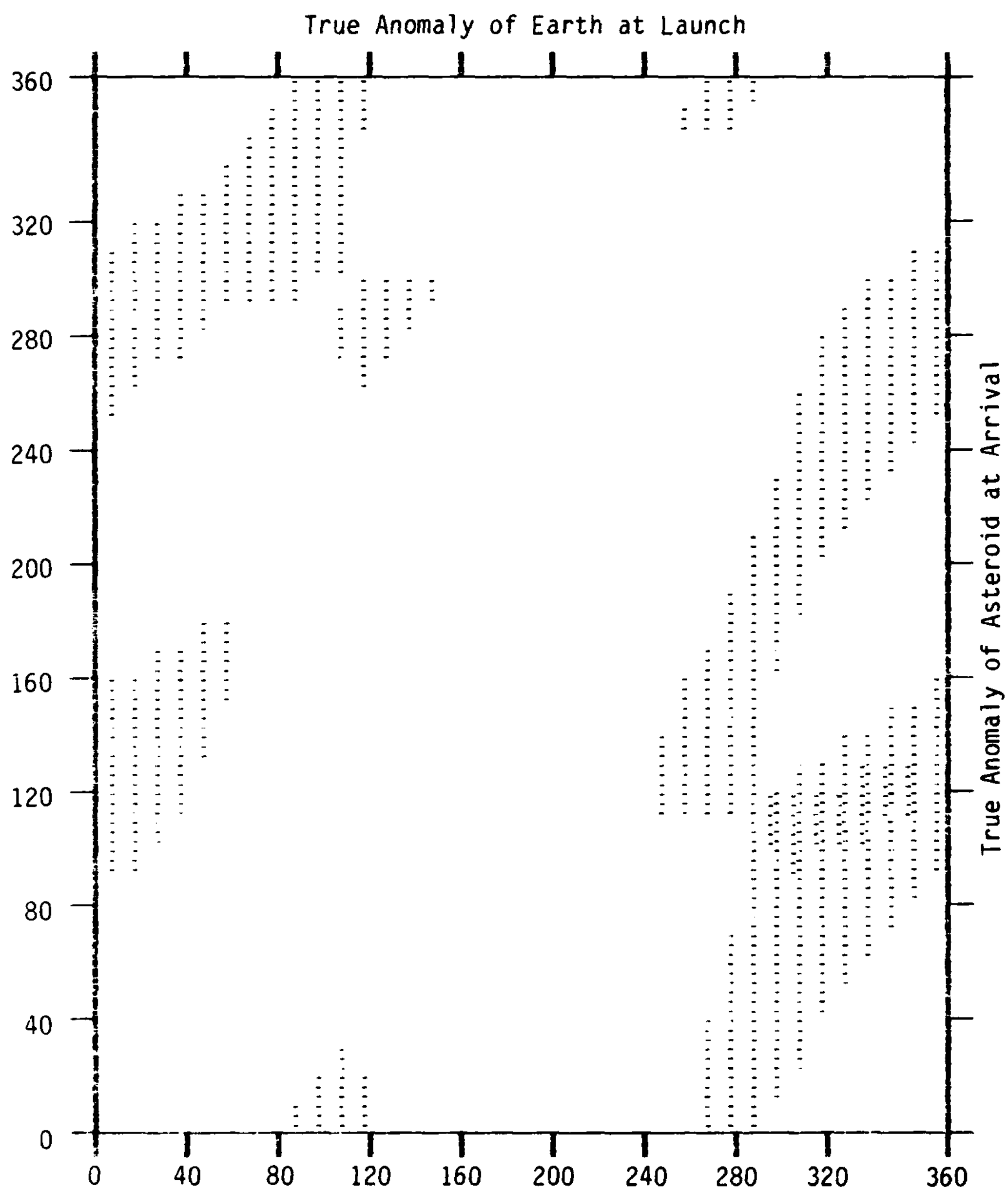
(Grey Scale: Black ≤ 4.5 km/s, White > 20 km/s)

ASTEROID 1985PA



(Grey Scale: Black $\leq$ 4.5 km/s, White > 20 km/s)

ASTEROID 1985TB



(Grey Scale: Black ≤ 4.5 km/s, White > 20 km/s)

Appendix H.

Delta Vee Tabulated Values for 1985TB

(True anomalies in degrees, delta vee in km/s)

Arr®Dep	0	10	20	30	40	50	60	70	80	90	100	110
360	56.8	47.2	39.4	33.3	28.7	25.1	22.4	20.3	18.8	18.0	18.0	19.4
350	46.1	38.4	32.7	28.4	25.1	22.5	20.5	18.9	17.7	17.1	17.6	19.7
340	36.7	31.4	27.6	24.6	22.4	20.5	19.0	17.8	16.9	16.5	17.3	20.5
330	29.5	26.1	23.7	21.8	20.2	19.0	17.9	17.0	16.3	16.1	17.2	22.1
320	24.2	22.3	20.8	19.6	18.7	17.9	17.1	16.5	16.0	15.8	17.2	25.6
310	20.6	19.5	18.7	18.1	17.6	17.1	16.7	16.3	15.9	15.7	17.3	37.1
300	18.3	17.8	17.5	17.2	17.1	17.0	16.9	16.7	16.5	16.0	17.2	48.5
290	17.0	16.9	16.9	17.1	17.4	17.7	18.2	18.8	19.9	23.3	20.9	17.4
280	16.5	16.8	17.2	17.9	18.8	20.1	22.1	26.3	38.6	54.9	19.2	17.4
270	16.8	17.4	18.4	19.9	22.0	25.4	31.7	46.1	60.8	29.6	19.5	18.5
260	17.7	18.9	20.8	23.5	27.8	35.2	49.2	71.2	37.7	24.4	20.1	19.6
250	19.3	21.4	24.6	29.4	37.3	50.2	67.7	42.2	29.1	22.7	20.7	20.8
240	21.8	25.2	30.4	38.3	50.2	65.0	44.3	32.4	25.4	22.2	21.4	22.0
230	25.5	30.8	38.7	49.8	62.8	45.0	34.3	27.4	23.7	22.2	22.2	23.2
220	30.9	38.7	49.3	56.4	44.7	35.0	28.5	24.7	22.8	22.5	23.1	24.4
210	38.5	48.8	54.7	43.9	34.9	28.8	25.0	23.1	22.5	22.9	24.1	25.8
200	48.5	53.1	42.7	34.2	28.4	24.9	23.0	22.3	22.5	23.5	25.2	27.2
190	51.8	41.3	33.0	27.5	24.2	22.4	21.7	21.9	22.8	24.4	26.4	28.8
180	39.8	31.5	26.2	23.1	21.4	20.8	21.0	21.9	23.4	25.4	27.9	30.7
170	29.8	24.7	21.7	20.1	19.6	19.9	20.8	22.2	24.2	26.7	29.7	32.9
160	22.8	20.1	18.7	18.3	18.6	19.5	21.0	22.9	25.4	28.5	31.9	35.7
150	18.4	17.2	16.9	17.2	18.1	19.6	21.6	24.1	27.1	30.7	34.8	39.2
140	15.7	15.5	15.9	16.8	18.2	20.2	22.6	25.7	29.4	33.7	38.6	43.7
130	14.4	14.8	15.7	17.0	18.9	21.3	24.3	28.1	32.6	37.8	43.4	49.2
120	14.1	14.9	16.2	17.9	20.2	23.1	26.8	31.4	36.9	43.1	49.6	57.7
110	14.6	15.8	17.3	19.5	22.3	25.9	30.5	36.2	42.9	50.2	60.3	54.3
100	15.9	17.3	19.3	21.9	25.4	30.0	35.8	43.0	50.9	62.7	55.7	48.8
90	17.9	19.7	22.2	25.6	30.1	36.0	43.5	52.0	64.7	56.4	48.6	42.2
80	20.5	23.1	26.4	30.9	36.9	44.6	53.6	62.7	56.6	47.9	41.1	36.3
70	24.1	27.6	32.2	38.4	46.3	55.6	65.3	56.4	46.9	39.9	34.9	31.3
60	28.7	33.6	40.1	48.3	57.8	68.0	56.1	46.2	39.0	33.7	29.9	27.4
50	34.8	41.6	50.2	59.8	70.4	56.1	46.0	38.5	32.8	28.8	26.1	24.5
40	42.6	51.5	61.4	67.9	56.4	46.4	38.5	32.5	28.2	25.2	23.4	22.4
30	52.2	62.5	68.4	56.8	47.0	38.9	32.6	28.0	24.7	22.6	21.3	21.0
20	63.0	68.8	57.2	47.5	39.4	33.0	28.2	24.7	22.2	20.6	19.8	20.0
10	68.9	57.2	47.6	39.6	33.3	28.5	24.9	22.2	20.3	19.1	18.8	19.5
0	56.8	47.2	39.4	33.3	28.7	25.1	22.4	20.3	18.8	18.0	18.0	19.4

Arr®Dep	120	130	140	150	160	170	180	190	200	210	220	230
360	22.6	28.2	37.0	50.0	67.1	76.1	60.0	47.5	38.5	32.3	27.8	24.5
350	24.5	33.0	46.6	65.6	75.4	58.5	46.0	37.6	31.9	27.9	25.0	22.7
340	27.9	41.6	62.8	74.1	55.9	43.6	35.9	30.8	27.4	25.0	23.1	21.8
330	34.6	58.4	72.1	51.8	39.9	33.2	29.1	26.4	24.5	23.2	22.2	21.6
320	50.8	68.9	45.8	35.1	29.7	26.8	24.9	23.7	23.0	22.5	22.2	22.2
310	63.3	37.2	29.1	25.8	24.2	23.3	22.8	22.6	22.7	22.9	23.3	24.0
300	26.0	22.9	22.0	21.8	21.8	22.0	22.4	22.9	23.6	24.4	25.6	27.3
290	18.5	19.3	20.0	20.7	21.4	22.3	23.2	24.3	25.6	27.3	29.4	32.2
280	18.2	19.1	20.1	21.2	22.4	23.6	25.1	26.7	28.8	31.2	34.3	38.3
270	19.1	20.1	21.2	22.6	24.1	25.7	27.7	30.0	32.7	35.9	39.8	44.9
260	20.3	21.4	22.8	24.4	26.3	28.4	30.8	33.6	36.8	40.7	45.5	51.4
250	21.7	23.0	24.6	26.5	28.7	31.2	34.0	37.2	41.0	45.5	50.7	56.3
240	23.1	24.7	26.6	28.8	31.3	34.1	37.3	40.8	44.9	49.5	54.4	59.1
230	24.7	26.5	28.7	31.2	34.0	37.0	40.3	44.0	48.2	52.4	56.6	47.5
220	26.2	28.4	30.8	33.6	36.5	39.7	43.1	46.8	50.6	51.8	47.0	42.1
210	27.9	30.3	33.0	36.0	39.1	42.4	45.7	49.2	50.7	46.4	42.0	37.6
200	29.6	32.4	35.3	38.5	41.8	45.1	48.4	49.8	45.9	41.9	37.9	34.1
190	31.6	34.6	37.9	41.2	44.6	48.0	49.2	45.6	41.8	38.1	34.5	31.1
180	33.8	37.2	40.7	44.3	47.9	49.0	45.5	41.9	38.4	35.0	31.7	28.7
170	36.5	40.3	44.1	47.9	49.1	45.7	42.2	38.8	35.5	32.3	29.4	26.7
160	39.8	44.0	48.1	49.5	46.2	42.8	39.4	36.1	33.0	30.1	27.4	24.9
150	43.8	48.4	50.7	47.0	43.5	40.1	36.8	33.7	30.8	28.1	25.6	23.3
140	48.8	52.6	48.6	44.6	40.8	37.5	34.3	31.4	28.7	26.2	24.0	22.0
130	55.1	50.5	46.0	41.8	38.1	34.9	31.9	29.2	26.8	24.6	22.6	20.8
120	52.5	47.3	42.6	38.5	35.2	32.2	29.6	27.2	25.1	23.3	21.6	20.1
110	48.3	43.0	38.6	35.2	32.2	29.6	27.4	25.5	23.9	22.4	21.2	20.1
100	42.9	38.3	34.7	31.8	29.4	27.4	25.8	24.4	23.3	22.5	22.0	21.9
90	37.4	33.9	31.1	28.8	27.1	25.8	24.8	24.2	23.9	24.1	25.0	27.2
80	32.7	30.0	28.0	26.6	25.6	25.0	24.9	25.2	26.1	28.0	31.7	38.7
70	28.7	26.9	25.8	25.1	25.0	25.3	26.2	27.9	30.7	35.4	43.6	57.5
60	25.7	24.8	24.4	24.6	25.4	26.8	29.1	32.8	38.4	47.4	61.1	78.1
50	23.6	23.4	23.9	25.0	26.9	29.8	34.1	40.5	50.2	63.8	79.6	55.1
40	22.2	22.8	24.2	26.4	29.8	34.6	41.8	52.0	65.8	74.2	57.7	43.6
30	21.5	22.9	25.3	29.0	34.4	42.1	53.0	67.1	75.2	59.5	46.0	35.9
20	21.3	23.7	27.5	33.2	41.4	53.0	67.8	75.9	60.5	47.4	37.7	30.7
10	21.6	25.3	31.1	39.8	52.1	67.8	76.2	60.7	47.9	38.6	31.9	27.0
0	22.6	28.2	37.0	50.0	67.1	76.1	60.0	47.5	38.5	32.3	27.8	24.5

Arr@Dep	240	250	260	270	280	290	300	310	320	330	340	350
360	21.9	20.1	18.8	18.1	18.5	20.2	23.9	29.8	38.4	49.5	62.1	68.8
350	21.1	19.8	19.1	19.1	20.0	22.6	27.6	35.8	47.2	61.0	68.1	55.7
340	20.8	20.2	20.0	20.6	22.4	26.2	33.4	44.7	59.6	66.9	53.8	44.0
330	21.2	21.2	21.8	23.2	26.2	32.1	42.7	58.4	64.9	51.0	41.1	34.2
320	22.6	23.3	24.8	27.5	32.7	42.2	57.9	62.3	47.6	37.6	31.2	27.0
310	25.2	27.0	30.0	35.0	43.6	58.5	59.6	44.5	34.3	28.2	24.6	22.2
300	29.6	33.0	38.2	46.3	59.7	57.7	43.0	32.4	26.3	22.7	20.5	19.2
290	36.0	41.1	49.0	60.5	56.7	43.3	32.5	25.7	21.8	19.5	18.2	17.4
280	43.4	50.8	60.4	56.4	44.8	34.2	26.7	22.0	19.3	17.8	16.9	16.5
270	51.5	59.5	56.1	46.2	36.5	28.7	23.3	19.9	18.0	16.9	16.4	16.4
260	58.1	64.7	47.3	38.7	31.1	25.2	21.2	18.7	17.3	16.6	16.5	16.9
250	61.9	47.8	40.4	33.4	27.5	22.9	19.8	18.0	17.1	16.8	17.1	17.9
240	47.8	41.4	35.2	29.6	24.9	21.4	19.0	17.7	17.2	17.3	18.1	19.5
230	41.9	36.5	31.3	26.8	23.1	20.2	18.5	17.6	17.6	18.2	19.7	22.0
220	37.2	32.6	28.4	24.7	21.7	19.4	18.2	17.9	18.4	19.7	22.0	25.5
210	33.5	29.6	26.1	23.0	20.6	18.9	18.2	18.4	19.5	21.7	25.3	30.7
200	30.4	27.1	24.2	21.7	19.7	18.5	18.4	19.3	21.3	24.8	30.2	38.2
190	28.0	25.1	22.6	20.6	19.0	18.3	18.8	20.6	24.0	29.4	37.6	48.2
180	25.9	23.5	21.3	19.6	18.5	18.3	19.7	22.8	28.2	36.7	48.0	50.7
170	24.2	22.0	20.2	18.8	18.0	18.5	21.1	26.3	35.1	47.7	50.0	38.2
160	22.7	20.7	19.1	18.0	17.6	19.1	23.6	32.7	47.0	49.6	36.4	27.8
150	21.3	19.6	18.2	17.2	17.3	20.2	28.8	45.3	49.4	34.1	25.5	20.8
140	20.1	18.6	17.3	16.4	16.9	22.9	41.4	49.0	30.9	22.6	18.7	16.6
130	19.2	17.7	16.5	15.6	16.6	32.3	47.1	25.9	19.2	16.4	15.1	14.5
120	18.7	17.3	16.1	15.0	16.0	37.7	18.7	15.7	14.5	13.9	13.6	13.7
110	19.2	18.5	18.2	19.9	19.1	13.7	13.6	13.4	13.3	13.3	13.5	13.9
100	22.5	25.1	35.5	52.5	16.7	13.8	13.5	13.5	13.6	13.8	14.2	14.9
90	32.5	46.0	61.1	28.9	16.4	14.2	14.0	14.1	14.3	14.8	15.5	16.5
80	52.7	75.4	39.0	23.1	16.2	14.6	14.5	14.8	15.4	16.2	17.2	18.7
70	76.6	46.3	30.0	20.6	16.1	14.9	15.1	15.7	16.6	17.8	19.4	21.4
60	51.4	35.7	25.5	19.3	16.0	15.2	15.7	16.7	18.1	19.8	22.1	25.0
50	40.2	29.8	22.9	18.4	16.0	15.6	16.4	17.8	19.8	22.2	25.4	29.5
40	33.3	26.1	21.2	17.9	16.1	16.1	17.2	19.2	21.8	25.2	29.6	35.3
30	28.8	23.6	20.0	17.6	16.4	16.7	18.2	20.8	24.3	29.0	35.0	42.8
20	25.6	21.9	19.2	17.5	16.8	17.5	19.6	22.9	27.6	33.9	42.1	52.0
10	23.4	20.8	18.8	17.6	17.5	18.6	21.4	25.8	32.1	40.6	51.1	62.8
0	21.9	20.1	18.8	18.1	18.5	20.2	23.9	29.8	38.4	49.5	62.1	68.8

Appendix I.

Time-of-Flight Tabulated Values for 1985TB

(True anomalies in degrees, time-of-flight in days)

Arr®Dep	0	10	20	30	40	50	60	70	80	90	100	110
360	180	217	270	310	335	346	347	340	330	317	303	288
350	234	292	336	363	375	376	371	361	348	334	319	301
340	324	371	398	410	410	404	393	380	367	353	336	315
330	420	446	455	452	442	429	414	399	385	372	355	331
320	511	514	505	489	471	452	433	415	400	389	376	345
310	593	575	552	525	498	472	448	427	410	400	400	351
300	669	634	596	558	523	490	460	432	410	396	423	315
290	740	690	640	593	548	508	471	436	402	369	455	344
280	809	745	686	630	578	532	489	452	425	462	486	401
270	872	802	734	671	615	565	524	493	521	533	518	445
260	931	855	783	716	659	611	571	539	585	588	553	485
250	979	903	829	763	705	658	618	650	645	633	588	525
240	1012	939	868	806	751	701	719	705	694	670	623	563
230	1028	961	895	836	782	784	762	745	730	700	653	600
220	1019	959	901	869	832	805	784	767	747	715	673	627
210	982	931	891	853	822	798	779	763	741	711	674	637
200	918	875	841	811	784	764	746	729	707	680	651	622
190	826	794	764	739	717	699	682	665	646	624	601	579
180	718	692	667	646	627	611	596	580	563	545	528	513
170	605	582	562	543	527	512	498	484	470	456	443	433
160	496	476	459	443	428	415	402	390	378	367	358	353
150	400	383	368	353	340	328	316	305	295	287	282	281
140	321	306	292	279	267	255	245	235	227	221	220	223
130	258	244	231	219	208	197	188	180	174	171	173	182
120	208	196	183	172	162	152	144	137	134	135	142	330
110	170	158	147	136	126	118	111	107	106	112	304	302
100	139	128	118	108	99	92	87	86	91	282	280	273
90	115	105	95	87	79	74	72	76	262	260	254	244
80	96	87	78	71	65	63	67	70	243	237	227	226
70	80	72	65	60	58	62	61	228	222	212	228	242
60	68	61	56	55	59	55	214	208	211	233	245	250
50	59	54	53	56	53	202	197	217	240	251	255	253
40	53	52	54	195	193	187	225	250	262	264	262	256
30	51	52	189	186	192	233	262	277	280	276	269	261
20	49	185	182	198	242	275	293	299	296	288	278	268
10	183	179	206	253	291	312	321	320	312	302	290	277
0	180	217	270	310	335	346	347	340	330	317	303	288

Arr®Dep	120	130	140	150	160	170	180	190	200	210	220	230
360	273	257	243	229	215	201	189	176	164	152	140	128
350	282	263	246	230	213	199	185	172	159	147	134	122
340	292	269	249	227	211	196	182	168	155	141	128	116
330	301	273	243	226	209	194	179	164	150	136	123	109
320	305	264	243	225	208	191	176	161	146	131	117	103
310	288	263	242	224	207	190	174	158	142	127	112	97
300	286	266	247	228	210	191	174	156	140	124	108	93
290	310	283	260	237	216	196	176	158	141	124	109	96
280	347	310	279	252	228	205	185	166	148	133	121	112
270	385	340	304	273	246	222	201	183	167	156	143	118
260	424	375	336	302	274	250	229	214	202	179	152	132
250	465	416	374	341	313	290	273	262	229	198	176	163
240	508	460	420	388	363	344	333	296	260	235	220	215
230	550	507	471	444	423	410	385	342	313	297	292	627
220	585	548	519	497	484	479	447	413	395	616	688	771
210	603	575	554	540	534	538	532	510	667	727	796	866
200	596	577	564	559	562	575	596	685	733	788	842	893
190	562	550	545	548	558	578	663	701	743	786	827	860
180	503	498	500	508	526	608	638	670	705	737	765	787
170	428	430	437	452	533	556	582	609	636	659	679	692
160	353	359	372	451	470	491	514	537	558	575	589	596
150	286	297	423	419	411	429	449	469	486	499	508	512
140	233	391	387	379	368	378	396	413	427	437	442	442
130	359	356	348	337	323	341	357	372	382	389	391	388
120	327	320	309	296	300	317	331	342	349	352	351	348
110	295	284	272	273	289	303	313	320	323	322	319	314
100	263	251	255	271	283	291	297	298	297	294	290	283
90	232	246	260	270	275	278	278	276	272	267	261	253
80	242	254	260	262	263	261	259	255	250	243	235	226
70	250	253	253	252	249	246	242	236	230	222	214	204
60	251	249	246	243	238	233	228	221	214	206	196	187
50	250	245	240	235	230	224	217	209	201	192	183	169
40	250	244	237	231	224	217	208	200	191	179	169	158
30	253	245	236	228	220	211	202	192	182	171	160	149
20	258	247	238	228	217	207	197	186	175	164	153	141
10	264	252	240	228	216	204	193	181	169	157	146	134
0	273	257	243	229	215	201	189	176	164	152	140	128

Arr®Dep	240	250	260	270	280	290	300	310	320	330	340	350
360	116	105	94	83	73	64	55	48	44	45	43	184
350	110	98	87	75	65	55	47	42	42	40	188	183
340	103	91	79	68	57	48	41	39	36	193	189	259
330	96	84	71	60	49	41	37	32	201	211	295	370
320	89	76	64	52	43	38	31	211	241	346	432	486
310	83	70	58	48	42	32	221	277	406	510	572	594
300	80	68	60	48	38	231	309	461	588	667	697	692
290	85	76	59	49	241	334	497	647	753	804	811	784
280	94	76	65	251	357	521	684	814	892	926	910	865
270	100	88	286	389	543	707	849	949	1009	1024	996	940
260	120	116	434	573	729	868	979	1058	1101	1103	1066	1005
250	158	491	616	757	889	995	1078	1136	1164	1158	1116	1053
240	558	669	794	912	1005	1081	1138	1182	1201	1188	1145	1086
230	725	831	930	1013	1075	1121	1164	1195	1204	1190	1150	1094
220	858	942	1009	1060	1093	1125	1153	1174	1179	1163	1128	1077
210	931	986	1026	1053	1070	1087	1108	1123	1123	1108	1077	1032
200	937	967	989	1001	1006	1016	1030	1041	1040	1024	996	959
190	886	904	911	913	913	918	929	935	933	917	893	863
180	801	808	808	804	799	802	809	815	810	796	774	747
170	699	700	696	688	681	681	688	690	685	671	654	629
160	599	595	587	577	568	568	574	575	568	559	538	516
150	510	504	493	480	469	470	475	475	473	456	437	418
140	438	429	417	402	388	391	396	401	387	371	354	337
130	383	373	360	343	323	331	345	334	319	303	287	272
120	341	331	320	303	274	303	293	278	263	249	235	221
110	308	300	292	284	234	248	241	230	218	206	194	182
100	276	267	254	225	203	204	200	193	183	173	162	150
90	243	232	208	192	178	175	171	164	156	146	136	125
80	215	204	183	169	158	154	148	142	133	124	115	105
70	194	176	163	152	142	136	130	123	115	107	98	89
60	171	160	148	138	129	122	115	108	100	92	84	76
50	158	147	136	126	117	110	102	95	87	80	72	65
40	147	137	126	116	107	99	91	84	76	69	62	56
30	138	128	117	107	98	89	81	74	66	60	54	51
20	130	120	109	99	89	80	72	65	58	52	49	50
10	123	112	101	91	81	72	63	56	50	47	48	47
0	116	105	94	83	73	64	55	48	44	45	43	184

Appendix J.

True Anomaly verses Date from the Year 1985 to 2020 for the Asteroid 1985TB

Julian Date	70	80	90	00	10	20	30	40	50	60
2446070.5	217	218	220	221	222	224	225	227	228	230
2446170.5	232	234	236	238	240	242	244	247	249	252
2446270.5	255	258	262	265	269	274	279	284	289	296
2446370.5	303	310	319	328	337	347	358	8	19	28
2446470.5	38	46	54	61	68	74	79	84	89	93
2446570.5	97	100	104	107	110	112	115	117	119	122
2446670.5	124	126	127	129	131	132	134	136	137	138
2446770.5	140	141	142	144	145	146	147	148	149	150
2446870.5	152	153	154	155	156	157	157	158	159	160
2446970.5	161	162	163	164	165	165	166	167	168	169
2447070.5	170	170	171	172	173	174	174	175	176	177
2447170.5	177	178	179	180	181	181	182	183	184	184
2447270.5	185	186	187	188	188	189	190	191	192	192
2447370.5	193	194	195	196	197	197	198	199	200	201
2447470.5	202	203	204	205	206	207	208	209	210	211
2447570.5	212	213	214	216	217	218	219	221	222	224
2447670.5	225	227	228	230	232	233	235	237	239	241
2447770.5	244	246	249	252	254	258	261	265	269	273
2447870.5	278	283	288	295	301	309	317	326	336	346
2447970.5	356	6	17	27	36	45	53	60	67	73
2448070.5	78	84	88	92	96	100	103	106	109	112
2448170.5	114	117	119	121	123	125	127	129	131	132
2448270.5	134	135	137	138	140	141	142	143	145	146
2448370.5	147	148	149	150	151	152	153	154	155	156
2448470.5	157	158	159	160	161	162	163	164	164	165
2448570.5	166	167	168	169	169	170	171	172	173	173
2448670.5	174	175	176	177	177	178	179	180	180	181
2448770.5	182	183	184	184	185	186	187	187	188	189
2448870.5	190	191	191	192	193	194	195	196	196	197
2448970.5	198	199	200	201	202	203	204	205	206	207
2449070.5	208	209	210	211	212	213	214	215	217	218
2449170.5	219	221	222	223	225	226	228	230	231	233
2449270.5	235	237	239	241	243	246	248	251	254	257
2449370.5	260	264	268	272	277	282	287	293	300	308
2449470.5	316	324	334	344	354	5	15	25	34	43
2449570.5	51	59	66	72	78	83	87	92	96	99
2449670.5	103	106	109	111	114	116	119	121	123	125
2449770.5	127	129	130	132	133	135	136	138	139	141
2449870.5	142	143	144	146	147	148	149	150	151	152
2449970.5	153	154	155	156	157	158	159	160	161	162
2450070.5	163	163	164	165	166	167	168	168	169	170
2450170.5	171	172	172	173	174	175	176	176	177	178
2450270.5	179	179	180	181	182	183	183	184	185	186

Julian Date	70	80	90	00	10	20	30	40	50	60
2450370.5	186	187	188	189	190	190	191	192	193	194
2450470.5	195	195	196	197	198	199	200	201	202	203
2450570.5	204	204	205	206	207	209	210	211	212	213
2450670.5	214	215	216	218	219	220	222	223	225	226
2450770.5	228	229	231	233	235	237	239	241	243	245
2450870.5	248	251	253	256	260	263	267	271	276	281
2450970.5	286	292	299	306	314	323	332	342	352	3
2451070.5	13	23	33	42	50	58	65	71	77	82
2451170.5	87	91	95	99	102	105	108	111	113	116
2451270.5	118	120	123	125	126	128	130	132	133	135
2451370.5	136	138	139	140	142	143	144	145	147	148
2451470.5	149	150	151	152	153	154	155	156	157	158
2451570.5	159	160	161	162	162	163	164	165	166	167
2451670.5	167	168	169	170	171	172	172	173	174	175
2451770.5	175	176	177	178	179	179	180	181	182	182
2451870.5	183	184	185	186	186	187	188	189	190	190
2451970.5	191	192	193	194	194	195	196	197	198	199
2452070.5	200	201	201	202	203	204	205	206	207	208
2452170.5	209	210	212	213	214	215	216	217	219	220
2452270.5	221	223	224	226	227	229	231	232	234	236
2452370.5	238	240	243	245	247	250	253	256	259	263
2452470.5	266	271	275	280	285	291	298	305	313	321
2452570.5	330	340	350	1	11	21	31	40	49	56
2452670.5	63	70	76	81	86	90	94	98	101	105
2452770.5	108	110	113	116	118	120	122	124	126	128
2452870.5	130	131	133	134	136	137	139	140	141	143
2452970.5	144	145	146	147	149	150	151	152	153	154
2453070.5	155	156	157	158	159	160	160	161	162	163
2453170.5	164	165	166	167	167	168	169	170	171	171
2453270.5	172	173	174	175	175	176	177	178	178	179
2453370.5	180	181	182	182	183	184	185	185	186	187
2453470.5	188	189	189	190	191	192	193	193	194	195
2453570.5	196	197	198	199	199	200	201	202	203	204
2453670.5	205	206	207	208	209	210	211	212	214	215
2453770.5	216	217	219	220	221	223	224	226	227	229
2453870.5	230	232	234	236	238	240	242	244	247	250
2453970.5	252	255	259	262	266	270	274	279	284	290
2454070.5	296	303	311	320	329	338	349	359	9	20
2454170.5	29	39	47	55	62	69	75	80	85	89
2454270.5	93	97	101	104	107	110	113	115	117	120
2454370.5	122	124	126	128	129	131	133	134	136	137
2454470.5	139	140	141	143	144	145	146	147	148	150
2454570.5	151	152	153	154	155	156	157	158	159	159

Julian Date	70	80	90	00	10	20	30	40	50	60
2454670.5	160	161	162	163	164	165	166	166	167	168
2454770.5	169	170	170	171	172	173	174	174	175	176
2454870.5	177	178	178	179	180	181	181	182	183	184
2454970.5	185	185	186	187	188	188	189	190	191	192
2455070.5	192	193	194	195	196	197	198	198	199	200
2455170.5	201	202	203	204	205	206	207	208	209	210
2455270.5	211	212	213	215	216	217	218	220	221	222
2455370.5	224	225	227	228	230	232	234	235	237	240
2455470.5	242	244	246	249	252	255	258	261	265	269
2455570.5	273	278	283	289	295	302	310	318	327	337
2455670.5	347	357	8	18	28	37	46	54	61	68
2455770.5	74	79	84	89	93	97	100	103	107	109
2455870.5	112	115	117	119	121	123	125	127	129	131
2455970.5	132	134	135	137	138	140	141	142	144	145
2456070.5	146	147	148	149	150	151	152	154	155	155
2456170.5	156	157	158	159	160	161	162	163	164	165
2456270.5	165	166	167	168	169	170	170	171	172	173
2456370.5	173	174	175	176	177	177	178	179	180	180
2456470.5	181	182	183	184	184	185	186	187	188	188
2456570.5	189	190	191	192	192	193	194	195	196	197
2456670.5	197	198	199	200	201	202	203	204	205	206
2456770.5	207	208	209	210	211	212	213	214	216	217
2456870.5	218	219	221	222	223	225	226	228	230	231
2456970.5	233	235	237	239	241	244	246	249	251	254
2457070.5	257	261	264	268	273	277	282	288	294	301
2457170.5	308	317	325	335	345	355	6	16	26	35
2457270.5	44	52	60	66	73	78	83	88	92	96
2457370.5	100	103	106	109	112	114	117	119	121	123
2457470.5	125	127	129	130	132	134	135	137	138	139
2457570.5	141	142	143	145	146	147	148	149	150	151
2457670.5	152	153	154	155	156	157	158	159	160	161
2457770.5	162	163	164	164	165	166	167	168	169	169
2457870.5	170	171	172	173	173	174	175	176	176	177
2457970.5	178	179	180	180	181	182	183	183	184	185
2458070.5	186	187	187	188	189	190	191	191	192	193
2458170.5	194	195	196	196	197	198	199	200	201	202
2458270.5	203	204	205	206	207	208	209	210	211	212
2458370.5	213	214	215	217	218	219	220	222	223	225
2458470.5	226	228	229	231	233	235	237	239	241	243
2458570.5	246	248	251	254	257	260	264	268	272	276
2458670.5	281	287	293	300	307	315	324	333	343	353
2458770.5	4	14	24	34	43	51	58	65		

Appendix K.

True Anomaly verses Date from the Year 1985 to 2020 for the Earth

Julian Date	70	80	90	00	10	20	30	40	50	60
2446070.5	2	12	22	33	43	53	63	73	83	93
2446170.5	102	112	122	131	141	151	160	170	179	189
2446270.5	198	208	218	227	237	247	256	266	276	286
2446370.5	296	306	316	326	336	346	357	7	17	27
2446470.5	37	47	57	67	77	87	97	107	117	126
2446570.5	136	146	155	165	174	184	193	203	213	222
2446670.5	232	241	251	261	271	281	291	301	311	321
2446770.5	331	341	351	1	12	22	32	42	52	62
2446870.5	72	82	92	102	112	121	131	141	150	160
2446970.5	169	179	188	198	207	217	227	236	246	256
2447070.5	266	275	285	295	305	315	326	336	346	356
2447170.5	6	16	27	37	47	57	67	77	87	97
2447270.5	106	116	126	136	145	155	164	174	183	193
2447370.5	202	212	222	231	241	251	260	270	280	290
2447470.5	300	310	320	330	341	351	1	11	21	31
2447570.5	42	52	62	72	82	91	101	111	121	130
2447670.5	140	150	159	169	178	188	197	207	217	226
2447770.5	236	246	255	265	275	285	295	305	315	325
2447870.5	335	345	356	6	16	26	36	46	56	66
2447970.5	76	86	96	106	116	125	135	145	154	164
2448070.5	173	183	192	202	211	221	231	240	250	260
2448170.5	270	280	290	300	310	320	330	340	350	0
2448270.5	11	21	31	41	51	61	71	81	91	101
2448370.5	111	120	130	140	149	159	168	178	187	197
2448470.5	206	216	226	235	245	255	265	274	284	294
2448570.5	304	314	324	335	345	355	5	15	26	36
2448670.5	46	56	66	76	86	96	105	115	125	135
2448770.5	144	154	163	173	182	192	201	211	221	230
2448870.5	240	250	259	269	279	289	299	309	319	329
2448970.5	339	350	360	10	20	30	40	51	61	71
2449070.5	81	90	100	110	120	129	139	149	158	168
2449170.5	177	187	196	206	216	225	235	244	254	264
2449270.5	274	284	294	304	314	324	334	344	354	5
2449370.5	15	25	35	45	55	65	75	85	95	105
2449470.5	115	124	134	144	153	163	172	182	191	201
2449570.5	210	220	230	239	249	259	269	279	288	298
2449670.5	309	319	329	339	349	359	9	20	30	40
2449770.5	50	60	70	80	90	100	109	119	129	139
2449870.5	148	158	167	177	186	196	205	215	225	234
2449970.5	244	254	263	273	283	293	303	313	323	334
2450070.5	344	354	4	14	24	35	45	55	65	75
2450170.5	85	95	104	114	124	133	143	153	162	172
2450270.5	181	191	200	210	220	229	239	249	258	268

Julian Date	70	80	90	00	10	20	30	40	50	60
2450370.5	278	288	298	308	318	328	338	349	359	9
2450470.5	19	29	39	49	60	70	79	89	99	109
2450570.5	119	128	138	148	157	167	176	186	195	205
2450670.5	214	224	234	243	253	263	273	283	293	303
2450770.5	313	323	333	343	353	4	14	24	34	44
2450870.5	54	64	74	84	94	104	114	123	133	143
2450970.5	152	162	171	181	190	200	209	219	229	238
2451070.5	248	258	268	277	287	297	307	318	328	338
2451170.5	348	358	8	19	29	39	49	59	69	79
2451270.5	89	99	108	118	128	137	147	157	166	176
2451370.5	185	195	204	214	224	233	243	253	262	272
2451470.5	282	292	302	312	322	332	343	353	3	13
2451570.5	23	33	44	54	64	74	84	93	103	113
2451670.5	123	132	142	152	161	171	180	190	199	209
2451770.5	219	228	238	247	257	267	277	287	297	307
2451870.5	317	327	337	347	358	8	18	28	38	48
2451970.5	58	68	78	88	98	108	118	127	137	147
2452070.5	156	166	175	185	194	204	213	223	233	242
2452170.5	252	262	272	282	292	302	312	322	332	342
2452270.5	352	2	13	23	33	43	53	63	73	83
2452370.5	93	103	113	122	132	142	151	161	170	180
2452470.5	189	199	208	218	228	237	247	257	267	276
2452570.5	286	296	306	316	327	337	347	357	7	17
2452670.5	28	38	48	58	68	78	88	98	107	117
2452770.5	127	136	146	156	165	175	184	194	203	213
2452870.5	223	232	242	252	261	271	281	291	301	311
2452970.5	321	331	342	352	2	12	22	32	43	53
2453070.5	63	73	83	92	102	112	122	131	141	151
2453170.5	160	170	179	189	198	208	217	227	237	246
2453270.5	256	266	276	286	296	306	316	326	336	346
2453370.5	357	7	17	27	37	47	57	67	77	87
2453470.5	97	107	117	126	136	146	155	165	174	184
2453570.5	193	203	212	222	232	241	251	261	271	281
2453670.5	291	301	311	321	331	341	351	1	12	22
2453770.5	32	42	52	62	72	82	92	102	111	121
2453870.5	131	140	150	160	169	179	188	198	207	217
2453970.5	227	236	246	256	265	275	285	295	305	315
2454070.5	325	336	346	356	6	16	27	37	47	57
2454170.5	67	77	87	97	106	116	126	135	145	155
2454270.5	164	174	183	193	202	212	221	231	241	251
2454370.5	260	270	280	290	300	310	320	330	340	351
2454470.5	1	11	21	31	41	52	62	72	81	91
2454570.5	101	111	121	130	140	150	159	169	178	188

Julian Date	70	80	90	00	10	20	30	40	50	60
2454670.5	197	207	216	226	236	245	255	265	275	285
2454770.5	295	305	315	325	335	345	355	6	16	26
2454870.5	36	46	56	66	76	86	96	106	116	125
2454970.5	135	145	154	164	173	183	192	202	211	221
2455070.5	231	240	250	260	270	279	289	299	309	320
2455170.5	330	340	350	0	10	21	31	41	51	61
2455270.5	71	81	91	101	110	120	130	139	149	159
2455370.5	168	178	187	197	206	216	226	235	245	255
2455470.5	264	274	284	294	304	314	324	335	345	355
2455570.5	5	15	25	36	46	56	66	76	86	95
2455670.5	105	115	125	134	144	154	163	173	182	192
2455770.5	201	211	220	230	240	249	259	269	279	289
2455870.5	299	309	319	329	339	349	360	10	20	30
2455970.5	40	50	60	70	80	90	100	110	120	129
2456070.5	139	149	158	168	177	187	196	206	215	225
2456170.5	235	244	254	264	274	284	294	304	314	324
2456270.5	334	344	354	5	15	25	35	45	55	65
2456370.5	75	85	95	105	115	124	134	143	153	163
2456470.5	172	182	191	201	210	220	230	239	249	259
2456570.5	269	278	288	298	308	318	329	339	349	359
2456670.5	9	20	30	40	50	60	70	80	90	100
2456770.5	109	119	129	138	148	158	167	177	186	196
2456870.5	205	215	224	234	244	254	263	273	283	293
2456970.5	303	313	323	333	344	354	4	14	24	34
2457070.5	45	55	65	75	85	94	104	114	124	133
2457170.5	143	153	162	172	181	191	200	210	219	229
2457270.5	239	248	258	268	278	288	298	308	318	328
2457370.5	338	348	359	9	19	29	39	49	59	69
2457470.5	79	89	99	109	119	128	138	148	157	167
2457570.5	176	186	195	205	214	224	234	243	253	263
2457670.5	273	283	293	303	313	323	333	343	353	3
2457770.5	14	24	34	44	54	64	74	84	94	104
2457870.5	113	123	133	142	152	162	171	181	190	200
2457970.5	209	219	229	238	248	258	267	277	287	297
2458070.5	307	317	328	338	348	358	8	18	29	39
2458170.5	49	59	69	79	89	99	108	118	128	137
2458270.5	147	157	166	176	185	195	204	214	223	233
2458370.5	243	253	262	272	282	292	302	312	322	332
2458470.5	342	353	3	13	23	33	43	54	64	74
2458570.5	84	93	103	113	123	132	142	152	161	171
2458670.5	180	190	199	209	218	228	238	247	257	267
2458770.5	277	287	297	307	317	327	337	347		

Appendix L.

Comparison of Computed Ephemerides with Almanac Data

HORB2050.PAS — ASTRONOMICAL ALMANAC 1985

Ephemeris for >>> 1 Ceres <<<

DD MM YEAR		Right Ascen	Declination	Right Ascen	Declination
26 07 1985	....	7 29 16.8	25 07 00	7 29 31.3	25 06 32
28 07 1985	....	7 33 07.3	25 02 41	7 33 21.6	25 02 13
30 07 1985	....	7 36 57.4	24 58 06	7 37 57.4	24 57 37

Ephemeris for >>> 2 Pallas <<<

DD MM YEAR		Right Ascen	Declination	Right Ascen	Declination
25 12 1985	....	6 00 57.4	-33 01 17	6 00 18.4	-33 00 51
27 12 1985	....	5 59 11.6	-32 55 39	5 58 32.9	-32 54 56
29 12 1985	....	5 57 27.0	-32 47 47	5 56 48.7	-32 46 48

Ephemeris for >>> 3 Juno <<<

DD MM YEAR		Right Ascen	Declination	Right Ascen	Declination
02 03 1985	....	12 44 48.3	-01 01 02	12 44 26.9	-00 58 35
04 03 1985	....	12 43 38.4	-00 44 28	12 43 16.5	-00 41 57
06 03 1985	....	12 42 24.4	-00 27 30	12 42 02.1	-00 24 57

Ephemeris for >>> 4 Vesta <<<

DD MM YEAR		Right Ascen	Declination	Right Ascen	Declination
10 09 1985	....	15 15 36.9	-14 53 30	15 15 44.7	-14 54 17
12 09 1985	....	15 19 09.3	-15 12 47	15 19 17.4	-15 13 33
14 09 1985	....	15 22 44.1	-15 31 51	15 22 52.4	-15 32 37

Projection to Year 2000

Ephemeris for >>> 2 Pallas <<<

DD MM YEAR		Right Ascen	Declination	Right Ascen	Declination
02 01 2000	....	08 09 56.4	-29 50 14		
03 01 2000	....	08 09 20.1	-29 49 41		
04 01 2000	....	08 09 42.6	-29 48 37	(not published)	
05 01 2000	....	08 08 42.6	-29 48 37		
06 01 2000	....	08 07 23.8	-29 44 50		

Appendix M.

Orbital Elements of Selected Solar System Objects

Name	Epoch (+2440000.)	Inclin. (deg)	Argue. of Perigee (deg)	Long. Ascend. Node (deg)	Semi-Major Axis (AU)	Eccen-tricity	Mean Anomaly (deg)
1985 JA	6200.5000	36.73616	288.7836	232.012	1.64345	0.32013	288.784
1985 PA	6300.5000	55.92239	311.4754	147.371	1.42205	0.30137	263.249
1985 TB	6432.5546	27.02800	66.8280	23.390	2.61136	0.57289	000.000
1982 DB	5647.7650	1.42009	157.8281	314.082	1.48932	0.36017	0.0000
1982 XB	6000.5000	3.87314	16.6825	74.5784	1.33767	0.44688	266.0197
1943 Ant.	6400.5000	8.70253	338.1039	245.785	1.43019	0.25591	128.210
1982 BB	6400.5000	20.94324	253.6499	129.2868	1.40682	0.35475	182.2862
1982 DV	6000.5000	5.92683	349.1949	218.2229	2.03324	0.45667	315.4827
1982 FT	6000.5000	20.38330	234.5135	348.3263	1.77421	0.28380	16.9277
1982 RA	6400.5000	32.97512	53.2491	339.4558	1.57466	0.28372	194.8538
1982 RB	6000.5000	24.99579	158.5766	158.4470	2.10193	0.39487	263.0551
1982 TA	6000.5000	12.11626	118.5972	10.0439	2.30298	0.77104	187.4833
1982 YA	6000.5000	34.57320	143.6389	269.1622	3.70673	0.69725	98.2681
1983 LB	6000.5000	25.39914	220.1517	80.9363	2.29142	0.47869	128.4299
1983 LC	6000.5000	1.51906	184.6915	159.0668	2.63152	0.70933	101.2684
1983 RB	6000.5000	19.42719	114.8082	168.8844	2.22334	0.50700	141.7119
1983 RD	6000.5000	9.51734	192.9485	173.4031	2.09011	0.48667	129.0814
1983 SA	6000.5000	30.77923	316.6039	350.0285	4.23072	0.71457	97.2714
1983 TB	6400.5000	22.02894	321.6679	265.0462	1.27132	0.89017	205.4340
1983 TF	5620.5000	7.83661	121.0523	10.4301	1.34276	0.38708	283.8069
1983 VA	6400.5000	16.23778	11.6840	76.8703	2.61068	0.69170	167.1080
1984 KB	6000.5000	4.63662	334.8782	170.5624	2.22103	0.76228	61.6257
1984 KD	6400.5000	13.61579	203.5824	81.8423	2.19758	0.54087	155.5356
1984 QA	6400.5000	9.91826	54.8267	152.0450	0.98963	0.46838	181.1960
1 Mercury	6280.5000	7.00578	29.0864	48.3492	0.38710	0.20564	230.6966
2 Venus	6280.5000	3.39475	54.8432	76.7188	0.72332	0.00681	256.0015
3 Earth	6280.5000	0.00185	107.9279	354.9000	1.00002	0.01669	208.8982
4 Mars	6280.5000	1.85078	286.3834	49.6025	1.52372	0.09331	140.6931
5 Jupiter	6280.5000	1.30465	275.1670	100.4667	5.20266	0.04808	301.3302
6 Saturn	6280.5000	2.48456	339.0884	113.7135	9.55775	0.05121	141.1483
7 Uranus	6280.5000	0.77457	102.2384	74.0564	19.27850	0.04662	75.3400
8 Neptune	6280.5000	1.76881	230.1408	131.8112	30.25704	0.00752	271.57664
9 Pluto	6280.5000	17.13148	114.1751	110.4183	39.60047	0.25128	353.66898

Sources: 1985JA (3:1)
1985PA (3:2) Planets (20:E3)
1985TB (3:3) Others (14:4)

Bibliography

1. Bate, Robert R. and others. Fundamentals of Astrodynamics. New York: Dover Publications Inc., 1971.

2. Battin, Richard H. Astronautical Guidance. New York: McGraw-Hill Book Company, 1964.

3. Central Bureau For Astronomical Telegrams. Circular No. 4132, and Minor Planet Circulars No. 9830 and No. 10034, Smithsonian Astrophysical Observatory. Cambridge, Massachusetts.

4. Cochran, Anita L. and Edwin S. Barker. "Minor Planet 1983TB: A Dead Comet? " Icarus, 59: 296-300 (August 1984).

5. Danby, J.M.A. and T.M. Burkhardt. "The Solution of Kepler's Equation, I," Celestial Mechanics, 31: 95-107 (May 1983).

6. Farquhar, Robert W. and David W. Dunham. "Earth-Return Trajectory Options for the 1985-86 Halley Opportunity," AIAA Paper No. 81-0314, presented at the AIAA 20th Aerospace Sciences Meeting, Orlando, Florida,11-14 January 1982.

7. Fox, James H. "Ephemeris for Comets and Minor Planets," 80 Microcomputing, 45: 90-98 (October 1983).

8. French, James R. and Neal D. Hulkower. "Exploration of the Asteroids," Journal of the British Interplanetary Society, 36: 167-171 (April 1982).

9. Hahn, Gerhard and Hans Rickman. "Asteroids in Cometary Orbits," Icarus, 61: 417-442 (March 1985).

10. Helin, Eleanor F. and E. M. Shoemaker. "The Palomar Planet-Crossing Asteroid Survey, 1973-1978," ICARUS, 40: 321-328 (December 1979).

11. Helin, Eleanor F. and others. "The Discovery of 1982 DB, the Most Accessible Asteroid Known," Icarus, 57: 42-47 (January 1984).

12. Hulkower, N.D. and others. "Optimum Two-Impulse Transfer for Preliminary Interplanetary Trajectory Design," Journal of Guidance, 7: 458-461, (1984).

13. Kaplan, Marshall H. Modern Spacecraft Dynamics and Control. New York: John Wiley and Sons, 1976.

14. Lau, C.O. and N.D. Hulkower. "On the Accessibility of Near-Earth Asteroids," AAS Paper No. 85-352 presented at the AAS/AIAA Astrodynamics Specialist Conference, Vail, Colorado, 12-15 August 1985.

15. Levin, B.J. and others. "Farmington Meteorite: A Fragment of an Apollo Asteroid," Icarus, 28: 307-324 (November 1976).

16. McFadden, Lucy A. and others. "Mineralogical-Petrological Characteristics of Near-Earth Asteroids," Icarus, 59: 25-40 (July 1984).

17. Ross, David J. and Neal D. Hulkower. "Optimal Orbital Transfers and Prime Rib Curves for Mission Design," AIAA Paper No. 81-0313, presented at the AIAA AIAA Aerospace Science Meeting, St. Louis, Missouri, 12-15 January 1981.

18. Stancati, Michael L. and John K. Soldner. "Near-Earth Asteroids: A Survey of Ballistic Rendezvous and Sample Return Missions," AAS Paper No. 81-185 presented at the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, Nevada, 3-4 August 1981.

19. United States Naval Observatory. Almanac for Computers 1985. Washington: Nautical Almanac Office, 1984.

20. United States Naval Observatory. The Astronomical Almanac for the Year 1985. Washington: Nautical Almanac Office, 1984.

21. Van Houton, C. J. and others. "The Palomar-Leiden Survey of Faint Minor Planets: Conclusion," Icarus, 59: 1-19 (July 1984).

22. Wetherill, George W. "Apollo Objects," Scientific American, 240: 54-65 (March 1979).

23. Wilkening, Laurel L. "Asteroid Missions: Goals for Elemental Abundance Analysis," Icarus, 40: 434-438 (December 1979).

24. Williams, J.G. "Determining Asteroid Masses from Perturbations on Mars," Icarus, 57: 1-13 (January 1984).

25. Yamada, Yoshiro. "The Origin of Apollo-Amor Objects," Journal of the British Interplanetary Society, 35: 459-461 (October 1982).

Vita

Major Philip W. Somers was born on 24 October 1948 in Summerside, Prince Edward Island, Canada. He graduated from Kensington Regional High School in 1966 and joined the Royal Canadian Air Force as a student at Le College Militaire Royal de Saint-Jean, Quebec. In 1969, he transferred to the Royal Military College of Canada, Kingston, Ontario and graduated in 1971 with a Bachelor of Science degree. In 1972, Major Somers received his pilot's wings from the Canadian Forces Flying Training School in Cold Lake, Alberta. He served two tours as a tactical helicopter pilot, in Petawawa, Ontario and Lahr, West Germany. In 1977-78, he attended the Canadian Forces Aerospace Systems Course in Winnipeg, Manitoba. He returned to flying, on the Argus and Aurora anti-submarine patrol aircraft in Greenwood, Nova Scotia. In 1982, Major Somers was transferred to the Maritime Proving and Evaluation Unit where he was a pilot and project officer on the Aurora. In 1984, he entered the Graduate Space Operations program at the Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, Ohio. Major Somers has been assigned to a staff officer position at the North American Aerospace Defense Command at Colorado Springs, Colorado.

Permanent address: 1075 Allegheny Drive,
Colorado Springs, CO
80919

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A163 916

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GSO/AA/85D-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering — 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	

12. PERSONAL AUTHOR(S)
Philip W. Somers, Major, Canadian Armed Forces

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1985 December	145

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Asteroids, Intercept Trajectories, Computer, Rendezvous Spacecraft, Space Navigation, Space Exploration
22	03		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: AN ANALYSIS OF THE ACCESSIBILITY OF EARTH-APPROACHING ASTEROIDS

Thesis Advisor: Dr. William E. Wiesel

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER 16 JAN 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. William E. Wiesel	22b. TELEPHONE NUMBER (Include Area Code) 513-255-4476 — 22c. OFFICE SYMBOL AFIT/ENY

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

The purpose of this ~~paper was~~ to analyze the accessibility of Earth-approaching asteroids using a computer program that was practical to run on a microcomputer. This analysis employs techniques that can easily be adapted to find optimal trajectories for a variety of orbital intercept applications.

The mathematical analysis was adapted from recently-developed algorithms that were designed to run on main frame computers using extensive software libraries and data resources. The computer program developed for this paper was designed to operate on an IBM PC equipped with an 8087 math co-processor chip. Programming was done in Turbo PASCAL Version 3.0 which supports the 8087 mathematical capabilities. The program was designed to be self-contained except for data files of orbital elements. The program was also designed to operate efficiently and quickly while retaining much of the accuracy found on the main frame implementations. Only nonperturbed Keplerian motion was modelled. Every effort was made to ensure the program was as flexible as possible. Any object in the solar system in heliocentric orbit can be used as either the departure or arrival body. Orbital element data files are included for all the planets, several periodic comets, all the recently-discovered Earth-approaching asteroids, and many of the main belt asteroids. This flexibility permits not only rendezvous missions to be calculated, but can just as easily handle fly-by trajectories and return-to-Earth missions.

END

FILMED

4-86

DTIC